Universidad Carlos III de Madrid

Department of Telematic Engineering

Telecommunications Engineering

Master Thesis

Learning Analytics Visualizations of Student-Activity Time Distribution for the Open Edx Platform

Author: Héctor Javier Pijeira Díaz

Supervisor: Pedro Muñoz Merino

Leganés, February 9, 2015

Universidad Carlos III de Madrid

Proyecto fin de carrera: **Learning Analytics Visualizations of Student-Activity Time Distribution for the Open Edx Platform**

Autor: **Héctor Javier Pijeira Díaz**

Tutor: **Pedro Muñoz Merino**

El Tribunal

Presidente: Carlos García Rubio

Vocal: Telmo Zarraonandia Ayo

Secretario: Antonio de la Oliva

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 11 de febrero de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de _____.

_____	_____	_____
Presidente	Vocal	Secretario

To my best and greatest teachers in life,
my biggest supporters and
the people I love the most: my parents.
Mom, dad, this is for you.

Acknowledgments

This section starts with a colossal, yet short of what they deserve, acknowledge to my best lifetime teachers, my parents. I owe them everything, including my ethos and my personal values, my culture of effort, dedication and discipline. In the direct production of the project I thank my father for assisting me with his LaTeX knowledge and looking for me a number of bibliographic resources for the introductory and state of the art chapters way bigger than the amount I could address.

I would like to thank Pedro José Muñoz Merino, my supervisor for this project, for his guidance, advice, availability and patience. It has been a pleasure working with him.

The biggest acknowledge in technical terms goes to my colleague Javier Santofimia Ruiz for sharing the know how he was learning with his hands-on work, concretely: shared a test Django application, the steps to enable tracking logs at edX, the pertinence of using a newly created settings file instead of modifying edX's default, the addition of tables to the database using Django models and his clarifications on opaque keys. He was always willing to help and made meaningful technical contributions to this project.

Thanks to José Antonio Ruipérez Valiente for sharing his experience working in the learning analytics field for other platform, as well as his advice on how to tackle certain technical issues.

I would like to credit Ariel Díaz de Armas with having demonstrated to me how to implement AJAX functionality and the debugging potential of Chrome browser through its Developer Tools.

Thanks to Dr. Javier Calzada Prado, who was my teacher for the subject "Conocimiento libre y aprendizaje en la web" (Free knowledge and learning on the web) for showing me beyond intuition and isolated resources I had used, with a systematic approach, the impressive potential of the virtual world for learning. From him I first heard of Open Educational Resources (OER) and Coursera, just to cite two examples.

To María Julián Mateos for sharing the edX installation guide she was preparing.

Projects are undertaken by human beings and therefore social relations and human emotions are not foreign to them. For the encouragement and emotional support I would like to thank my family and all of my Cuban and Spanish friends, the old and new ones, no matter where in the world they are, they have managed to show me they care and that they are there for me. I am proud of my family and thankful to my friends for having entered my life.

Abstract

MOOCs are one of the current trending topics in educational technology. They surged with the vision of a democratization in education worldwide by removing some access barriers. As every technology, MOOCs have promoters and detractors but truth is, they are an invaluable source of data related to student interaction with courses and their resources as has been available never before. This data is susceptible to shed light on the learning process in this online environment and potentially influence in a positive way the learning outcomes. Students can be presented with visual, friendly information that enable them to reflect on their performance and gain awareness of their own learning style based on data beyond intuition. Teachers can be given the same metrics augmented with student aggregates for their courses. Thus, they can tune their pedagogical approach and resource quality for the better. In this context, Open edX is one of the most prominent MOOC platforms. However, its learning analytics support is low at present. This project extends the learning analytics support of the Open edX platform by adding new six visualizations related to time on video and problem modules, namely: 1) video time watched, 2) video and 3) problem time distributions, 4) video repetition profile, 5) daily time on video and problem and 6) distribution of video events. The main technologies used have been Python, Django, MySQL, JavaScript, Google Charts and MongoDB.

Keywords: MOOC, Open edX Platform, learning analytics, visualizations, student activity, educational technology

Resumen

Los MOOCs están de moda en lo que se refiere a tecnología educativa. Surgieron con la visión de remover algunas barreras de acceso en aras de la democratización de la educación en cada rincón del mundo. Como toda tecnología, tienen sus promotores y detractores, pero lo cierto es que constituyen una valiosa fuente de datos como no ha habido antes en lo que respecta a la interacción de los estudiantes con estos cursos y sus recursos. Estos datos pueden ayudarnos a entender el proceso de aprendizaje en estos entornos. Tienen además el potencial de influir positivamente en los resultados del aprendizaje. Se puede presentar a los estudiantes una información visual fácil de entender, que les permita reflexionar sobre su rendimiento y ganar conciencia de su estilo de aprendizaje a partir de los datos, más allá de lo que les pueda indicar la intuición. Las mísmas métricas se pueden poner a disponibilidad de los profesores, en conjunto con valores agregados de la clase. De esta manera, los profesores pueden ajustar el enfoque pedagóico del curso y mejorar la calidad de los recursos. En este contexto, Open edX es una de las plataformas proveedoras de MOOCs más prominentes. Sin embargo, tiene todavía poco soporte para analítica del aprendizaje. Este proyecto extiende ese soporte al incorporar seis visualizaciones nuevas sobre tiempo en vídeos y problemas, específicamente: 1) tiempo visto de vídeo, distribución de tiempo en 2) vídeos y 3) problemas, 4) perfil de repetición de vídeo, 5) tiempo diario en vídeos y problemas y 6) distribución de eventos de vídeo. Las principales tecnologías usadas son: Python, Django, MySQL, JavaScript, Google Charts y MongoDB.

Palabras clave: MOOC, Plataforma abierta edX, analítica del aprendizaje, visualizaciones, actividad del estudiantado, tecnología educativa

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Education is the most powerful weapon which you can use to change the world", said the great Nelson Mandela. This *weapon* has deeply evolved over time, marked by new discoveries, devices and technologies. Nowadays, a huge offer of (free) virtual courses, even from world-leading universities, is available for everyone everywhere in the world with an internet connection. The steps willing students take to interact with these resources leave a virtual trail (rich amount of data), which is a valuable source of information for themselves, their teachers or course authors and pedagogical researchers, who are all of them the main stakeholders in the learning process. This project deals with the design and implementation of useful visualizations that can help teachers and students during the learning process.

1.1 Educational resources from books to MOOCs

To understand where do we go let us start by learning where do we come from. Historical notes follow.

Technological expectations in the pre-online era

In ancient times knowledge was only acquired through own experience and oral transmission from one to another(s). The invention of paper helped conserve what people knew, or at least believed to know. But it was still difficult to propagate from both technical and political/religious viewpoints. Books or written knowledge in general had to be reproduced by hand, mostly by the hands of monastic scribes.

This would end in the 15$^{\text{th}}$ century with the introduction of revolutionary printing techniques by Johannes Gutenberg. Book reproduction became increasingly easier and thus they multiplied and consolidated their monopoly as knowledge-storage entities. Education became inconceivable without books.

In the 20$^{\text{th}}$ century however, new technologies would succeed one another awaking expectations of rethinking education. Let us examine some examples following [30]. In 1922, Thomas Edison said [10, p. 9]: "I believe that the motion picture is destined

to revolutionize our educational system and that in a few years it will supplant largely, if not entirely, the use of textbooks". Nowadays, motion picture usage in education is perhaps almost exclusive to language schools and history of cinema students. In fact, the term motion picture itself is rarely used, being more likely referred to as movie or film. Concepts we tend to associate with leisure, far from formal education. Then it was the radio. One of the advocates of the educational possibilities offered by radio was Benjamin Darrow, who in 1929 founded the Ohio School of the Air, which broadcast courses in subjects such as art appreciation and history, for example. Three years later, Darrow published his book *Radio: The Assistant Teacher*, from which the following quote has been widely cited, perhaps because it encloses the expectations that came with radio:

> The central and dominant aim of education by radio is to bring the world to the classroom, to make universally available the services of the finest teachers, the inspiration of the greatest leaders...and unfolding world events which through the radio may come as a vibrant and challenging textbook of the air.

William Levenson took over the school direction after Darrow and in 1945 claimed: "the time may come when a portable radio receiver will be as common in the classroom as is the blackboard. Radio instruction will be integrated into school life as an accepted educational medium" [10, p. 19]. Decades later we now it didn't come that way. However, we are likely to recognize there is a similar idea today but with tablets instead of radios.

Television incorporated a new feature to the audio of radio: image. Third quarter of 20^{th} century could be considered as the golden age of instructional television, which according to [10, p. 29] was used following one of these three patterns:

1. Total instructional program presented by television teacher. The classroom teacher plays no role.
2. Supplemented television instruction. The classroom teacher is in charge of introduction, discussion and assignments related to the video lesson.
3. Television as a teaching aid. Instruction doesn't rely on television, which use is determined by the classroom teacher on the topic he/she consider it could be of help.

In the 1980s personal computers appeared and with them new possibilities. Seymour Papert, a professor with the Massachusetts Institute of Technology, was a pioneer in posing computer's potential to impact learning. Widely recognized as the seminal thinker about ways in which computers can change learning,[1] Papert he authored two books on this topic: "Mindstorms: Children Computers and Powerful Ideas" in 1980 and "The Children's Machine: Rethinking School in the Age of the Computer" in 1994, when there was a broader computer's adoption rate. In the preface to the latter, he wrote:

[1] Papert's page at the MIT http://web.media.mit.edu/ papert/

> Information technologies, from television to computers and all their combinations, open unprecedented opportunities for action in improving the quality of the learning environment, by which I mean the whole set of conditions that contribute to shaping learning in work, in school, and in play.

We were able to witness later that even better than computers, independently of their evolution, was the synergy created by the connections among them, computer networks, the biggest of which is the known-to-everyone Internet. If we are connected to the latter, the so called network of networks, we say we are online. The online world empowered the idea of sharing to an extent unthinkable before.

Learning in the virtual world: resources and technologies

The World Wide Web launching in 1992 was the starting point for open information resources to quickly spread as freely available. It marked the explosion of information quantity to the levels of overloading familiar to us today with the all the impact on quality it poses. Therefore, quality ranges from poor to awesome [2, p. 1].

Year 1994 saw the foundation of the first 100% online university in Catalonia, Spain [53]. It was the Universitat Oberta de Catalunya, which translates into English as Open University of Catalonia. The possibility was real for students to study online officially recognized courses.

In many universities became customary the use of web pages to post course resources, many of which entered in the category of what would be known as *learning objects*, a term coined in 1994 by Wayne Hodgins. The IEEE defined the concept as "any entity, digital or non-digital, which can be used, reused, or referenced during technology-supported learning" [7, p. 45]. According to The Wisconsin Online Resource Center, learning objects should be self-contained, i.e. self-explained knowledge bits, and modular, i.e. facilitating aggregation and reusability. For the sake of exemplification we could mention applets, animations, videos... For engineering students some of those applets made easier the understanding of physical principles or the visualization of complex mathematical functions. Repositories were created to store and share learning objects. Browsing through categories and areas of knowledge users could seek for those most suitable to their interests. Learning objects detractors complained of the trouble they posed for adaptation to fit the specific needs of different courses and students. [23]

The Massachusetts Institute of Technology (MIT) came up with the OpenCourseWare (OCW) Initiative in 2002. "The OCW concept is based on the philosophical view of knowledge as a collective social product and so it is also desirable to make it a social property", claims Prof. V.S. Prasad [54, p. 15]. Questions were raised automatically about the value of content. MIT's freely release of course resources pointed out a philosophy that the economic value for learners lies in faculty, learner interactions and accreditation rather than in academic content [51, p. 47]. Note that the adjective academic itself describes the quality expected from open courseware.

Often, they are the exact same material available to those on-campus students taking the subject, to the extent that according to UNESCO [54, p. 5] they should consist of course description, syllabus and calendar at a minimum, as well as one or a number of the following:

- lecture notes;
- demonstrations, simulations, illustrations, learning objects;
- reading materials;
- assessments; and
- projects.

Note from above that they serve as a container for learning objects. However, OCW are not normally intended to provide direct open learning support for students. In fact, UNESCO recommended [54, p. 21] to recognize the faculty as the primary user of OCW, regarding it as a teaching resource, which is that also for learners. In the same document, which was the Final Report for the *Forum on the Impact of Open Courseware for Higher Education in Developing Countries* held in Paris in 2002, UNESCO suggested both a name and a definition for the services of open courseware. The name was Open Educational Resources (OER) and the definition: "The open provision of educational resources, enabled by information and communication technologies, for consultation, use and adaptation by a community of users for non-commercial purposes" [54, p. 24]. It is important to underline here that when we talk of either Open Educational Resources or open courseware, we are speaking about the same idea. Though they are frequently interchangeable and interchanged, in principle open courseware is not the same as OpenCourseWare. The former was the generic name that preceded OER, while the latter is the name of the MIT initiative in that direction, which is the epitome of OER.

A number of projects stemmed from the OER movement. For the purpose of outlining their gists, they can be clustered in the following classes, according to [2, p. 15]:

1. Incubation of high-quality specialized open resources.
2. Capacity building in developing countries for effective use of OER.
3. Toward building a relevant research community.
4. Building awareness, voice, and understanding.
5. General software and middleware services infrastructure for creating, federating, and finding OER resources.

In the research point, for example, the Berkeley's Center for Studies in Higher Education at the University of California saw in OER an opportunity to study the use of web-based collections of open academic content.

In late 1990s started the development of learning management systems (LMS) to back conventional courses. They provided features such as discussion forums and live chats in addition to lectures, reading material and assignments. WebCT (Web

Course Tools), BlackBoard (1997) and Moodle (2002) were among the most popular and became fixtures of many campuses [51, p. 14]. LMS are typically a tool for a closed community, especially that of a faculty classroom.

Stephen Downes had pioneered the development of learning objects, built an LMS from scratch and together with his colleague George Siemens, defined the learning theory called connectivism [2]. This theory, in the own words of Siemens, "is the view that knowledge and cognition are distributed across networks of people and technology and learning is the process of connecting, growing, and navigating those networks" [51, p. 11]. Based on this vision, they mounted and taught an online course in learning theory in 2008. The 25-student course at the University of Manitoba, Canada, attracted the interest of over 2300 people who took the subject via online [39]. The course is widely considered to be the first of the so called Massive Open Online Courses (MOOCs), the well-known most recent tool for providing free high-quality courses.

1.2 Motivation

Education is one of the most sensitive aspects of any society. It is an invaluable tool for society to give their knowledge to the individual and receive the reward in the form of the sum of individual contributions. A give-and-receive pattern to build and preserve values as well as boost innovation, creativity, progress.

Traditional pedagogical theories centered around how could teachers best transfer knowledge to their students. Computer-enhanced learning environments have brought the focus on how to help students internalize the presented knowledge for them to be able to make sense of it and apply it effectively [2, p. 45].

The relatively young learning analytics fields inherits from visual information techniques and data mining to serve the purpose of learning about learning and empowering individual-level learning stakeholders such as teachers and students to make data-driven decisions. We believe in the potential that properly presented and interpreted data has for the learning outcomes. Learning analytics help teachers assess top concerns and achievements related to their students. Among the powerful features of learning analytics are prediction of results, recommendations of learning paths and visualization of the learning process.

According to UNESCO [50, p. 2], the interest in learning analytics concerns a broad variety of roles, such as: "researchers in education, leaders and policy-makers, educational practitioners, organisational administrators, instructional designers, product vendors, and [...] the learners themselves". The benefits of learning analytics at different levels [50, p. 7] are multiple:

- Detect students falling behind.
- Present learners with their learning profile for self-reflection.

[2] www.downes.ca

- Support administrative decision-making and institutional resource allocation.
- Shed light on organizational's successes and challenges.
- Enable timely response to challenges and therefore increment organizational productivity .
- Assist in assessing the value created by faculty activity.
- Help devise new academic approaches and pedagogical models.

In the beginning platforms implemented rudimentary facilities to capture learner activity data for analysis. They rarely went beyond exporting comma-separated value (CSV) files of raw data for teachers and researchers. Still, simple information is given to the students, mainly about their marks and progress, though e-learning systems are providing increasingly valuable reporting tools. This project is determined to make its contribution there.

Nowadays, MOOCs are one of the trending topic when it comes to education present and future. They are challenging the creativity of traditional teachers and inviting worldwide people with an Internet connection to learn topics and subjects they are interested in for either professional or personal purposes. They are also a source of data about student activity as it has never existed before. The bulk of data is neither human-addressable nor often human-readable. That is why we need to build information out of data to make it meaningful and interpretable.

In the MOOC scenario, the combination of opportunity and necessity has fostered the **learning analytics** field, whose object of study is the data generated by students on an online platform. MOOCs are unbeatable in collecting online learning data. The large number of events registered and the mass of students generating events make them a renewable source of raw data upon which learning analytics can build. The potential is exciting to equip MOOCs with tools and techniques that could timely, positively impact the learning process. Student-learning data empowers teachers to tune course structure and materials on the basis of reflecting on the observed progress and use of the resources [23, p. 3, 4, 6].

We have identified an opportunity of making a contribution in this direction to the MOOC provider platform edX. Why edX? Because it is an open-source platform since June 2013 that welcomes and encourages community participation. EdX offers appealing features to their students like a molecule editor, a protein builder, a circuit simulator, a code grader tool, discussion forums, integration with social networks, downloadable videos and their transcripts, video segment location from the transcript and peer grading, just to name a few. It has an awesome functionality designed to provide students with a memorable learning experience. Tens of universities all-around the world, non-governmental organizations, international institutions such as the International Monetary Fund (IMF), the World Economic Forum (WEF) and the Institute of Electrical and Electronics Engineers (IEEE), and world-class businesses like Johnson and Johnson, have built their instances of the Open edX platform, which gives an idea of its broad acceptance and fast-growing adoption.

EdX learning analytics support is in its early stages. A column graph to show progress is the unique visualization for edX's students. The graph displays the percentage of chapters' graded problems completed. For teachers, the "Analytics" section in the "Instructors" navigational tab for course staff only reads "No Analytics are available at this time". In other section, "Datadump", teachers are able to export student grades to a CSV file. Visualizations available for teachers are related to demographics, course enrollment and number of students entering the course on a weekly basis. It is evident that the Open edX platform can be extended in number and nature when it comes to visualizations for students and teachers as well. In fact, at edX a "big investment" in "improve learning analytics with visualizations" is planned, as claimed edX's Vice President of Strategic Partnerships, Johannes Heinlein, at a talk he gave in the University Carlos III of Madrid on September 23, 2014. Heinlein also stated that "edX provides the tracks and the wagons on top of the rails are up to the institutions". We want to build on those tracks and add value to the railway.

As of March 2014, the University Carlos III of Madrid, where this project comes from, adopted edX and is currently offering MOOCs on the platform. With more and more courses coming from more and more educational institutions the data flow does nothing but to multiply. We believe that a visual, human-friendly presentation of those data is the key to harness its potential for the knowledge transmission chain. It will enable awareness and reflection on personal performance on students side and the selection and/or creation of better methodological approaches on the teachers side, as well as the design of quality course resources that help convey knowledge more effectively.

1.3 Objectives

The main objective pursued in this project is the **development of a set of six visualizations displaying learning analytics indicators for the Open edX platform**. They will have a strong time orientation in contraposition to more traditional grade-driven metrics. These visualizations will not be foreign to edX but integrated into its navigational structure. The visualizations will be available for the roles of both teacher and student. However, to ensure compliance with edX privacy policy students will be shown only their own data in the visualizations, while teachers will be able to select the student whose data they want to visualize, or even class aggregates, typically the student average. The latter will be the default option.

This objective will be broken down in six sub-objectives, one per each visualization. Therefore, sub-objectives will be the design and implementation of the following visualizations:

1. video time watched,
2. video time distribution,
3. problem time distribution,

4. video repetition profile,
5. daily time on video and problem modules, and
6. distribution of video events within video length.

To fulfil our objective knowledge of a variety of web development assisting technologies is a must. Python programming language will be the key on server side and HTML with JavaScript capabilities on client side as standards guaranteeing compatibility and browser-independent navigation.

The objective is challenging in that edX is a relatively new platform with new features being often added that even involve architectural changes. It is constantly evolving as the number of community contributions and contributors grow as does the adoption rate by universities and educational institutions worldwide. Most changes keep backwards compatibility but there are also inevitable disruptive changes in the platform. Following the natural cycle of such dynamic system, the documentation is not in a mature state still. Moreover, it is sparse and scarce, making code reading the most reliable way of learning edX. The diversity of technologies edX uses makes it also a very complete exercise to work inside and for the platform.

A second and no less important objective is the **integration of our application with that of other colleagues to be released as a more complete and powerful learning analytics module for the Open edX platform**. This is not an independent project, but a joint effort stemming from of a broader educational technology project run from the Gradient laboratory. Gradient[3] is the e-learning laboratory inside the Telematic Applications and Services Group (GAST), as a part of the Department of Telematic Engineering, at the Carlos III University of Madrid. Gradient's research lines include, but are not limited to, technology enhanced learning (TEL), adaptation of learning environments and learning analytics, involving visualizations and recommenders. Our target is the visualization line and we are working together with other colleagues in complementary contributions.

The Open edX platform receives code contributions every day and innovation is non-stop [1, p. 32]. It is a collaborative project with a philanthropic mission in which we also believe. That means harmony is a pillar to be always present. The code should be easily integrated, avoid repetitions to comply with the DRY (Do not Repeat Yourself) principle, reuse existing functions and create just those that really add value. We should follow the guidelines and conventions of edX developers for Python in naming variables and functions, indentation, letter case and the likes. The same nomenclature should be used to avoid confusion between concepts or terms and keep the logic behind them.

We are willing to make a contribution to the community of edX users and developers. The project will be released as open source after its completion. As added benefits it will contribute a number of functions or methods that can be reused for other developers to help them accelerate their work.

[3] http://gradient.it.uc3m.es

1.4 Scheduling

The list of tasks necessary to carry out the project follows. Tasks are then represented in the Gantt chart of 1.1.

Task 00.- Project assignment

Task 01.- Review of HTML, Javascript, CSS.
Familiarization with Python programming language, GIT repositories, Google App Engine and Google Course Builder.
Install Google Course Builder.
Reading of papers on the field of Learning Analytics, especially linked to the Khan Academy.

Task 02.- Project platform change from Course Builder to the Open edX platform.

Task 03.- Enrollment for edX's Demo Course.
Familiarization with edX at user level.

Task 04.- Read edX's documentation.
Examine edX code and architecture, focusing on the learning analytics support.

Task 05.- Installation of edinsights, loghandlersplus and djeventstream.
Read their documentation.

Task 06.- Installation of Devstack (edX development stack). Several attempts in three computers.
Familiarization with Virtual Box and Vagrant.

Task 07.- Ubuntu installation.

Task 08.- Check technology under edX's progress graph.
Selection of a chart gallery for visualizations.

Task 09.- Familiarization with the Django web framework. Take its tutorial. Dummy tests.

Task 10.- Familiarization with MongoDB. documentation.

Task 11.- Familiarization with Mako template language and where it is used at edX.

Task 12.- Virtual machine reprovision after news of changes on edX platform.

Task 13.- Learn about edX databases, namely MySQL and MongoDB, and ways to access them.
Find where logs are stored and how they can be accessed.
Familiarization with MySQL Workbench tool and configuration to connect to the edX MySQL database.
Analogously for Robomongo and the Mongo database.

Task 14.- Create a Django application external to edX in charge of analyzing data. (Failed)

Task 15.- Create a Django application inside edX.

Task 16.- Contribute as second author to a paper for the Second International

Conference on Technological Ecosystems for Enhancing Multiculturality [44].

Task 17.- Interaction with the development platform as user to generate event logs.

Task 18.- Add a new tab to edX course navigation menu to show the visualizations.
Review of regular expressions.
Dummy tests on this tab using Mako templates, JSON, HTML and JavaScript code.

Task 19.- Apply data processing techniques to get data from logs and obtain the required analytics.
Implement the six visualizations for students: video time watched, video and problem time distribution, video repetition profile, daily time on video and problem and video events distribution within video length. At this point, visualizations had no aggregates, just personal information, as they were first made for the student role. API creation.

Task 20.- Make a slideshow to present the proposed visualizations to an edX staff member visiting our university, to receive feedback on our contribution.

Task 21.- Extend the visualizations for teachers, adding selectors and student aggregates as required.

Task 22.- Familiarization with AJAX techniques, jQuery library for JavaScript and Django support for AJAX.
Optimization of data sent from client to server as needed for the visualizations according to user selection using AJAX.

Task 23.- Code refinement and debugging of semantic errors.

Task 24.- Architecture adaptation to Celery (splitting up data processing and visualization).

Task 25.- Vagrant machine reprovision to update the platform to the Opaque-Keys architecture.
Code adaptation to be compatible with the latest platform version (some functions and data types were deprecated).

Task 26.- Write the report.

Days are not representative of the work load as time was not homogeneously allocated to them. In this sense we must distinguish several periods:

1. April 23 to September 3, 2013: time shared among several subjects and the project. In average, two hours a day.
2. September 4 to October 20, 2013: time shared between a partial-time internship at a telecommunications company and the project. Three hours in average devoted to the project on a daily basis.
3. October 21, 2013 to April 30, 2014: time shared between my last remaining subject, a full-time internship at another telecommunications enterprise and the project. In average, ten hours a week allocated to the project.

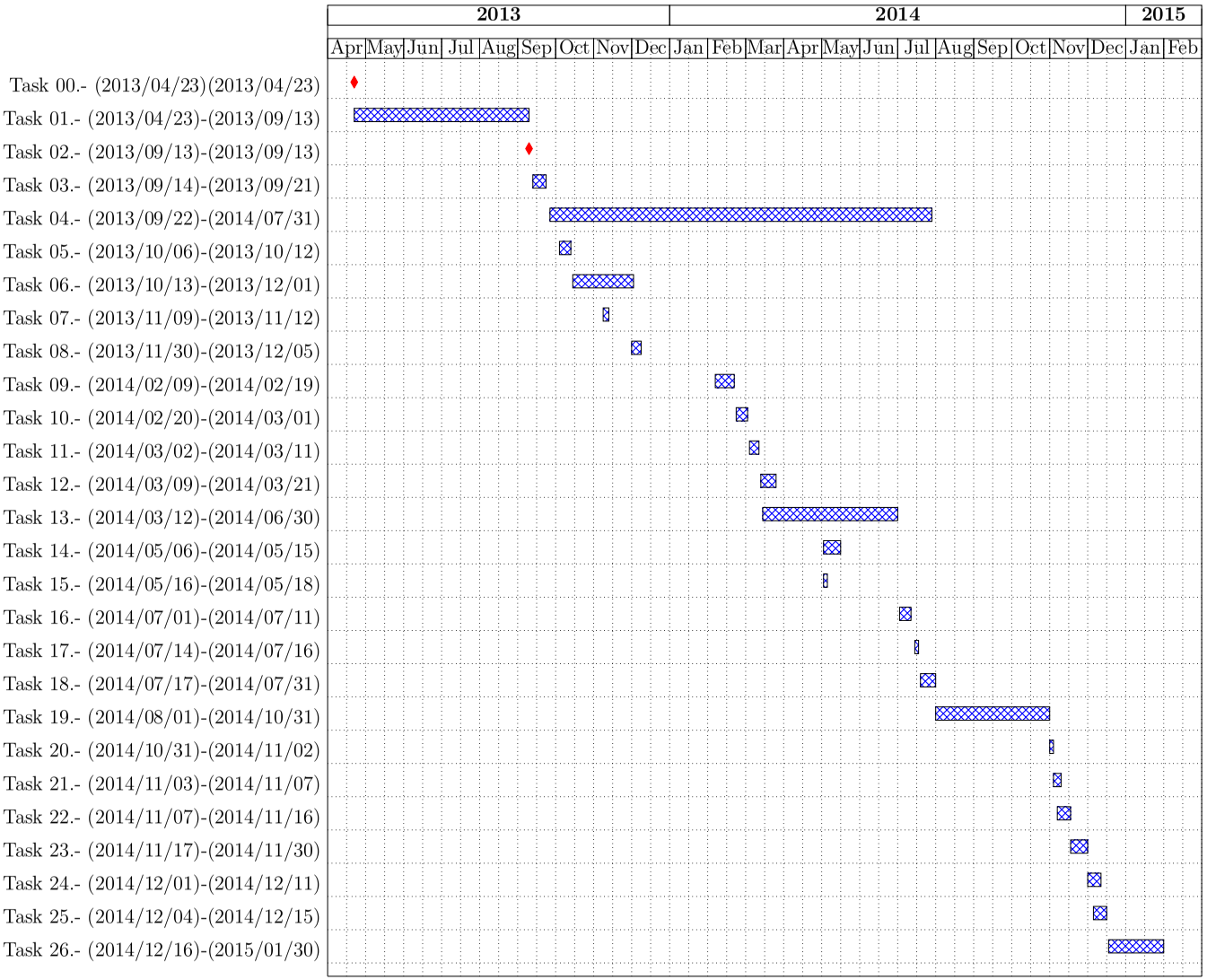

Table 1.1: Project scheduling in Gantt diagram.

4. May 1 to August 31, 2014: reduction in the internship working hours. Seventeen hours a week reserved for the project.
5. September 1 to project end: time shared between study of two foreign languages and the project, almost full dedication to the latter with an average of 38 weekly hours.

This is an a posteriori scheduling of the tasks carried out during the development of this project. E-mails, messages at the internal forum created for the project, notes, program logs and my human memory have been used to track the process and to have a planning as close as possible as it really was.

However, dates are often approximate as the end of some tasks and the beginning of the next ones are not that clear due to either their own nature or the subjectiveness of human mind. Some comments on tasks, challenges they posed, dates and landmarks follow.

On April 23, 2013, after a meeting with my tutor I was assigned the project. In the beginning it was aimed at contributing some learning analytics to Google Course Builder, an open source online education platform. So the first task was the review or learning of related technologies to be able to put hands to work. The review involved HTML, Javascript and CSS, which I had studied or exercised in at least a subject. New topics were the Python programming language (syntax, philosophy, guidelines), the GIT version control system, Google App Engine and Google Course Builder. The latter included also the installation of its application. In addition, I was recommended the reading of papers in the field of learning analytics, and some examples and applications to the Khan Academy.

On September 10, however, Google announced in its Research Blog that they were shifting their development efforts to join the Open edX Platform as a contributor [4]. That decision made us reconsider the platform object of our work, agreeing to contribute also to edX, the initiative led by the MIT and the Harvard University, and early supported by other world-class university, instead of Course Builder, whose responsible itself, Google, had decided to limit its development. In spite of this shift, we were still able to reuse some of the technologies learned or reviewed for the contribution to the Course Builder platform, especially Python, HTML and JavaScript.

On the developer level, understanding edX architecture, code organization and structure to being able to start effectively contributing to the platform was by far the most challenging, time-consuming task of the project. Needless to say, in this collaborative, developing platforms code is anything but static. Development platform installation was not either straightforward. It takes several hours and a number of attempts is usually necessary, following advice from questions asked on forum depending on the error faced. I had to install it several times and even with that gained experience every new time posed challenges.

[4] http://googleresearch.blogspot.com.es/2013/09/we-are-joining-open-edx-platform.html

The edX development stack is referred to as the devstack. The first devstack installation attempt was on my desktop PC with a processor Intel®Pentium®D CPU 925 3.00GHz. Installation failed as it did not support virtualization. The Department of Telematics gently assigned me a laptop for the time needed to carry out the project but it did not work properly either. So I decided to acquire a laptop with a compatible processor in early November. It shipped with Windows 8 and installing Ubuntu alongside it was troublesome due to some UEFI (Unified Extensible Firmware Interface) features. UEFI is a standard firmware interface for PCs designed to replace the BIOS (Basic Input-Output System). Dual boot was hard to achieve and even some restoration to factory settings was necessary. On November 23, the devstack was installed.

A few days later, I saw a special offer at the same store I had bought the laptop. There was another one, better in its technical characteristics including processor, and cheaper. So I decided to return the former and acquire the latter. The new one had a processor Intel®Core™i7-3612QM CPU @ 2.10GHz. The same process for dual operative system and devstack platform installation had to be repeated on this computer. As of December 1st, 2013, I had the platform installed for the second time.

Then came a several-month period of familiarization with different technologies, especially database-related. As of May, the project contribution was thought as an external Django application able to connect to the edX databases, process data and present the results in the form of visualizations. This approach failed as connection to edX databases from an external project was not achieved.

Having seen that edX's two main cores, the Learning Management System (LMS) and the Content Management System (CMS), were themselves Django projects and that the latter are comprised of many applications the next approach came clear: create a Django application internal to the LMS (where was our contribution interest as the CMS is more a course authoring tool, also called Studio). This way, our application would have the same permissions and accesses granted as the rest in the project. This proved to be the correct path.

1.5 Resources

For the development of the project is required on the hardware side a computer (PC or laptop) supporting Intel®Virtualization Technology (Intel®VT). Intel has a downloadable for free tool, the Intel Processor Identification Utility, to check if the processor in the system where it is launched supports it. At a minimum, the system should have 8 GB of free space as the VirtualBox Virtual Machine takes up several gigabytes. An Internet connection will also be necessary.

On the software side, the lists of programs or software technologies used follow, telling between required and optional. Nonetheless, those in the optional list (or equivalents) are strongly recommended as development facilitators. Versions are

included to provide specific identification of the tools this project was based on, they do not mean in any case that that versions are indispensable or even recommended.

Required tools:

- Ubuntu 12.04 operative system
- VirtualBox 4.3.20 (with Guest Additions 0.10.0)
- Vagrant 1.6.5
- NFS (Network File System) client
- A text editor (Kate was used)
- Django 1.4.16 web framework
- Python 2.7
- Google Charts
- Git 1.7.9.5

In principle, Ubuntu operative system is not mandatory as Windows and Mac OS installations are also reported to have been completed for some users in forums. As long as I am concern, I tried Windows installation several times without a successful result. However, Ubuntu makes installation easier (note the comparative adjective easier does not necessarily mean it was easy at all) as it ships with some required tools and often there is more documentation oriented to this operative system, particularly when it comes to development. This is an opinion and a personal experience.

Though I have used Kate editor to write Python code, Mako templates, HTML and Javascript functions, that is not even close to what would have been optimum. Developing environments for Python and built-in support for Django and web developing such as PyDev for Eclipse or PyCharm would have helped boost productivity and efficiency. Unfortunately, I learned about those tools too late.

Optional tools (recommended):

- MySQL Workbench 1.4.16
- Robomongo 0.8.4
- Chrome's Developer Tools 39.0.2171.99 (64-bit) (or equivalent in other browsers)

1.6 Report structure

To assist the readability of this document the content of each of the remaining chapters is outlined below.

The report continues with chapter 2, where related works are reviewed along with what are MOOCs, which types are there, what are the most popular platforms, paying especial attention to edX, of course, as the target of this project. The features edX has to offer, its current adoption level for universities and learners worldwide that had embraced the initiative, and where an opportunity for contribution has been identified.

Chapter 3 describes in detail what are the visualizations pursued in this project, what they have in common, why they have been chosen and for which purpose or what benefit could be obtained from them.

Once it is clear what we want to achieve, implementation is the next step. This is the purpose of chapter 4. All details are given so that the work is reproducible. What was necessary to do, which new functions where created and why, API details, problems faced, adopted solutions. This chapter is the project's core.

Results achieved are commented in chapter 5 as well as the tests they underwent to check requirement's fulfilment and that the solution worked in an expected, stable, secure and consistent way.

The report wraps up with a concluding chapter 6, to what extent objectives were fulfilled , what was achieved and what was not. Recommendations are also made to continue working in the line of the project, make improvements and detected contribution opportunities still waiting out there.

Chapter 2

State of the art

2.1 Massive Open Online Courses (MOOCs)

What is a MOOC?

The acronym MOOC for Massive Open Online Courses was coined in 2008 by David Cormier, who described them as "open, participatory, distributed, life-long networked learning" courses [8, min. 1:04]. The popularization of MOOCs, though, came three years later, in 2011, when professors Sebastian Thrun and Peter Norvig widely publicized they were going to offer online for free their Artificial Intelligence (AI) course at Stanford. Over 160,000 of 190 countries signed up for the ten-week course [11, 48], though about 23,000 went all the way through it [32, p. 26]. The spotlight came into MOOCs to the extent that for Siemens, citing a number of articles, newspapers, TV and radio programmes and blog posts, MOOCs was the higher education buzzword for 2012 [52].

MOOCs comprise a whole subject though less material is covered than in the equivalent college courses as they typically last four to ten or twelve weeks. Often, lecture segments come shorter and in larger numbers [34, p. 2]. They provide a coherent learning sequence, with integrated learning materials and formative assessment.

MOOCs represent a major shift in scale beyond learning objects and OER [23, p. 2]. They are significant in that they are a large public experiment exploring the impact of the internet on education. Some justifies the excitement around this courses both for their potential value and the quality of the players behind them.

It is not that clear what the differences between MOOCs and LMS are. LMS are considered the conceptual ancestors of MOOCs. Their birth philosophy differs in that the former were aimed at assisting a classroom, while the latter was intended to replace it. That was at least in the beginning since it seems nowadays expectations are shifting towards enhancing on-campus teaching.

MOOCs' free access and high-quality content features make them an object of desire for many people who registers for the courses. Few of them, however,

pass the courses. The high attrition rate was one of the early-identified patterns that troubled researchers. The first-generation MOOCs successful completion rate oscillates between 5 and 10 percent. In a course run by MIT for which over 150,000 students signed up, only 7,157 of them ended-up passing it [23, p. 5]. The ratio could appear disappointing. But is really 7,157 a low figure? It may be short of expectations though pretty high above on-campus students taking the subject. So, should it be considered a failure or a success? It concerns researchers who are on the line of establishing what could be an acceptable completion rate for MOOCs and what differences are to consider with respect to classroom courses regarding that threshold. Student demographics and motivation are key factors in the outcome.

Although these massive courses are destined to a worldwide, diverse audience, truth is that the average student in a MOOC is a young white American man with a bachelor's degree and a full-time job [48]. This is exemplified for the particular case of the edX platform in figure 2.1. To provide access is not enough in order to fulfill their mission.

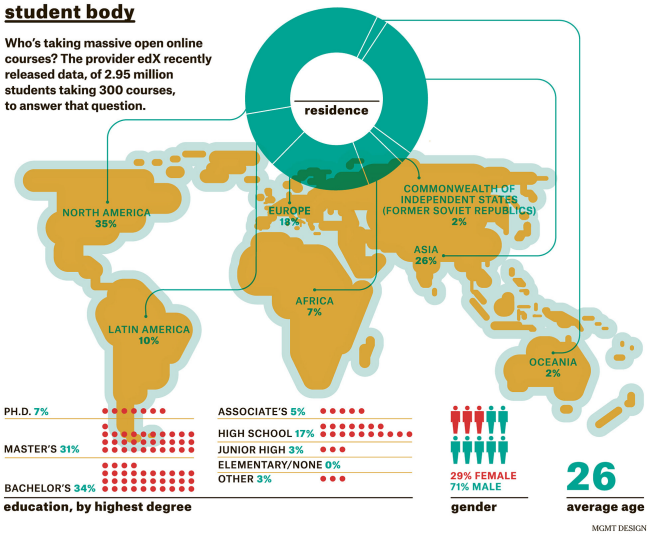

Figure 2.1: EdX platform demographics.

Source: The New York Times [48]

MOOCs are said to have disappointed many people who saw in them the promise of education democratization [55]. However, they are relatively new and we do not need to think of them as a panacea, but as a valuable opportunity we can of course take or reject. Research is ongoing on how we could get the most out of them and precisely this project is the contribution of tools to widen awareness of their use. The hype cycle chart in 2.2, developed by the American information technology research and advisory company Gartner, comes handy for the purpose of illustrating that MOOCs are just following a natural new technology life cycle. That is, after great

expectations appears a deflation point where the outlooks seem to have vanished. On the contrary, technology continues its way to maturity. Though the chart ends in the regular production stage, we need to mention a final phase is missing, namely, obsolescence.

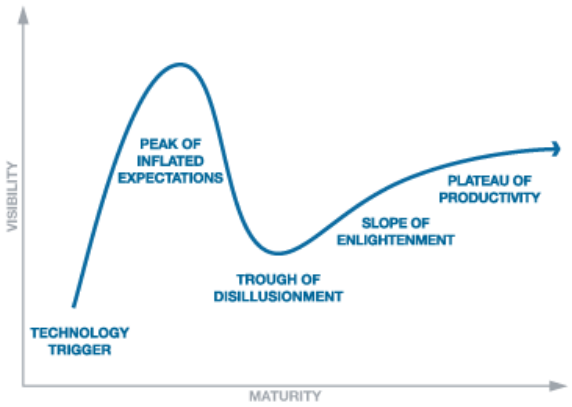

Figure 2.2: Five key phases of a technology's life cycle.

Source: Gartner

A joint working paper between the HarvardX Research Committee at Harvard University and the Office of Digital Learning at MIT [20], after studying a number of MOOC courses, arrived at conclusions like (just to name a few):

- there will be no grand unifying theory of MOOCs,
- measuring learning requires a greater investment in assessment and research, and
- open online courses are neither useless nor the salvation of higher-education.

What kinds of MOOCs are there?

The short answer is two: cMOOCs and xMOOCs.

Siemens make a distinction between what for him are two MOOC branches or offerings: connectivist MOOCs (cMOOCs) and traditional-learning MOOCs (xMOOCs, the x comes from the edX platform). In his view, "cMOOCs focus on knowledge creation and generation whereas xMOOCs focus on knowledge duplication". In my opinion, they do not have to be antagonists, but complementary, each having a function and covering a need. It is useful to have a tool to help build new knowledge, and it is important to have a tool to facilitate access to already created knowledge.

MOOC platforms

Currently the biggest MOOC platforms are Coursera and edX [48]. Other popular platforms are Udacity and Khan Academy.

The sound success of the course "Introduction to Artificial Intelligence" led Sebastian Thrun together with Stanford's colleagues to the foundation of the MOOC platform Udacity, a for-profit start up.

Coursera has also its origin at the Stanford University. It is an educational for-profit company founded by professors Andrew Ng and Daphne Koller. Most Coursera lectures have been recorded especially for its courses, often in a way that makes the lecturer seem to be speaking directly to the MOOC student viewer [34, p. 2].

Khan Academy was founded in 2006 by Salman Khan with the mission of providing "a free, world-class education for anyone, anywhere". Lectures are taught mainly by videos recorded in a hand-writing style, as you were watching the teacher writing on the blackboard.

EdX is the target of this project and we will discuss its features and characteristics in following sections. Video is the MOOC equivalent to the on campus class. This is so much so that for example, in edX, lectures are "often recorded in front of a class, complete with students walking in front of the camera as they arrive late, coughing, chatting with neighbours, or answering questions posed by the lecturer" [34, p. 2].

A partial reproduction of a table from [23] detailing the comparison of features between edX, Coursera and Google Course Builder platforms, appears in table 2.1.

2.2 Open edX platform

EdX is a MOOC platform originated from a joint partnership between Harvard University and the Massachusetts Institute of Technology in 2012. It is one of the most recent proposals in this realm [40, p. 1].

Open edX is the open-source educational platform developed by edX and its open source partners, including leading institutions.

The three overall goals of edX are [57]:

1. Expand access to quality education for everyone, everywhere.
2. Enhance teaching and learning on campus and online.
3. Advance teaching and learning through research.

In our project we will target the last two goals.

EdX welcoming interface is as shown in figure 2.3, with some personal data and configuration options in the left menu and the list of courses enrolled for at the center. Once the user enter the course the navigation starts at the Course Info tab by default on the navigation menu. This page is used to communicate courses updates and news to students. Courses at edX are under the Courseware tab. Here the user is presented the course chapters that often matches the work for a

Features	edX	Coursera	Google Course Builder
Video lectures			
Where are they stored?	YouTube	Coursera	YouTube
Quizzes integrated with video?	No	Yes	No
Discussion on video page?	Yes	No	No
Additional files and features	Subtitles	Subtitles files	Subtitles files
Quizzes			
Are there quizzes outside of videos?	Yes	Yes	Yes
Question types			
Multiple choice	✔	✔	✔
Short answer	✔	✔	✔
Numeric			
No. of attempts allowed	Limited	Limited	Unlimited
Discussion forums			
Can posts be rated?	Positive	Positive/negative	N/A
Grading and analytics			
Student's view of progress	Raw marks with graph	Raw marks	None
Teacher's view of progress	Unknown	Unknown	CSV* export Google analytics (CSV)

Table 2.1: Comparison between MOOC platforms [23, p. 4]

week. At the left menu there is an index-like course content with the chapters and learning sequences. By clicking on the desired sequence, it is recommended to take them in the order they come, the sequence appears as a horizontal navigation bar containing the different course resources. Learning sequences may contain video, explanations and problems in an impressive variety. There a number of problem types and ways in which they are graded, including self-assessment by comparing to correct or proposed solutions, peer-grading and machine-grading using artificial intelligence techniques. There are learning sequences to acquire knowledge, driven by videos and other sequences aimed at practicing what has been learned, driven by problems. EdX features interactive resources as it aims at providing a memorable learning experience.

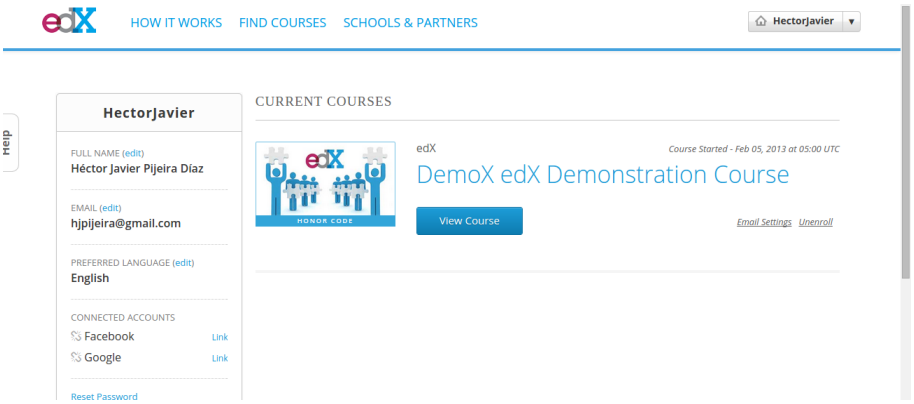

Figure 2.3: Screenshot from the live edX platform.

At the end of a course there are exams. By passing the exams students may earn an honor certificate as a souvenir of their achievement, or even a certificate from the institution providing the course. These are called verified certificates and depend on the type of enrollment and on the availability of this option for the course. Certificates are not for free and they are one of the features that research has shown drives course completion by the students taking these courses. EdX has granted over a hundred thousands certificates.

Currently at edX there are over 3 million users registered and 420 courses are offered. How were the beginnings?

Between October 15, 2012 and September 15, 2013 a number of 17 courses were released [20, p. 4, 10] from HarvardX and MITx. This marked the edX explosion in a fast-changing, rapid-growing period for the platform. There was a bulk of data generated by these first pack of courses, each of which accounted for 20 GB in average [20, p. 5].

Although they all fit under the course nomination, they were very diverse beyond content. The courses showed structural dissimilarities in design, duration, assessment policy and certification criterion. The first analytics considered were

certification percentage, gender ratios and mark histograms. Education, age, gender, nationality were among the demographics that centered early measures to gain an insight on who was taking the courses and what was his/her background. By the way, Spain was the sixth country by number of registrations with a 2.24% (16,926) and second only in Europe to the United Kingdom [20, p. 25]. All the more, Spain headed the table of certification rate statistic with a 13.74%.

According to their activity, users were classified following four mutually exclusive classes as: only registered, only viewed, only explored, explored and certified. The metrics showed that the percentage of active users, considering active by simply having signed in to the platform, dropped from a 50% in the first week of the course to a 5% in the tenth week [20, p. 30], in an exponential-like evolution. Much like the other MOOC platforms, edX faces challenges, but it is growing at an impressive pace and innovation is a constant at the platform.

When it comes to development, we would have to mention that edX is supported by an eclectic variety of technologies, which together with scarce and sparse documentation act as an entry barrier to developers. Although this is getting easier with time as documentation is being improved and more resources and sources appears as fellow developers familiarize with the platform and share their experience through papers, articles and forums.

EdX roadmap

In this section we review the features edX is working on to release in the coming months according to [57]. The public roadmap is divided into the following working areas:

- Teaching and Learning Tools;
- Mobile;
- Open edX and Platform Investments;
- edX.org;
- Professional Education;
- Data & Analytics; and
- Accessibility.

In the *Teaching and Learning Tools* line are planned the randomization of the problem collection, integration with collaborative and planning tools as Google Docs and Calendar respectively; reinforcement of course prerequisites and creation of prior-to-enrolling assessments.

Mobile goals are destined to "offer compelling and effective educational experiences on any device" and enabling "a more seamless transition between devices". Native applications for iPhone and Android platforms are scheduled to at least make possible to watch the videos, check course announcements and deliver handouts. In the middle to long run the table app is to come. Investments on the platform involve, but are not limited to, the organization of an Open edX Conference, the

creation of a website for the community and the activation of a service for platform notifications.

EdX site, *edX.org*, will be as mobile-friendly as possible, incorporate PayPal functionality in the payment flow and the dashboard interface revised.

Regarding professional education, the platform is to be enabled for making accounting and marketing reports and will offer organizations the possibility of massive purchase of courses concerning their interest.

Data & Analytics is the category where the contribution of this project fits. The main bet of edX with regard to this segment is on the launching of the edX Insights tool with special emphasis on student performance, behavior and demographic statistics. In the platform itself, edX is willing to "enhance data visibility and analysis capability" and they are said to "support contributions of new reports and analyses". We are very proud to do our bit here and we expect it will be useful for the community.

In the accessibility segment, the goal is to enhance "level of compliance with well established web accessibility guidelines". Here there is also very good news for future edX-community developers: the documentation will be enriched and modified "with a focus on practical guidance". This was perhaps the first hurdle we found while starting to carry out this project. Learning the platform's how-to by reading code was really time-consuming, yet rewarding in the long run.

2.3 Learning Analytics

In 2004, "a novel approach of using web log data generated by course management systems (CMS)" was presented in a system their authors called CourseVis [33]. It used information visualization techniques to build for example three dimensional bubble charts and heat maps to represent student's performance and participation in the discussion forums as initiators or repliers.

In 2010, a learning analytics application called *Moodog* [64] was designed an implemented to be integrated as a module to the Moodle CMS. It was the result of an interdisciplinary collaboration among computer science, psychology and pedagogical departments at the University of California. The tool uses Moodle logs to determine the student activity in the platform. It incorporated a "simple graphical interface" to present the results in a visual way.

They categorized the metrics [64, p. 3] using four groups:

1. Course summary: enrollment figures, number of resources, sessions per student, the students with more course interactions and the most popular course resource.
2. Per-student statistics: student accesses to course material, time spent, discussion threads initialized and follow-up messages posted on the forum.
3. Per-resource statistics: for each material the number of times is has been accessed and by who.

4. Time-based statistics: time and times on which students work on the course with various time resolution units available: week, day of the week and hour of the day.

That same year an approach to personalize exercises to the student level based on Contextualized Attention Metadata (CAM) [35] was presented. This data is generated by user interaction with resources in a certain context and its processing enables hints and problem level/characteristics be tailored to the student needs.

In 2012, the web-based visualization platform GLASS (Gradient's Learning Analytics System) [29] was released. It was not limited to but especially used to account for student activity in computer applications as part of their learning program. Thus, examples of parameters measured are daily events, use of bash command and the compiler, number of URLs visited and opening and closing of the text editor to write code (presumably).

In 2013, a module for incorporating new learning analytics visualizations to the Khan Academy platform was proposed, designed and implemented. It was named Learning Analytics Support in the Khan Academy (ALAS-KA) [42, 43] and included a number of visualizations like a video abandon pie chart, a time on exercises and video column chart, a use of platform bar chart, a video percentage visualization in a sliced donut chart and time distribution per topics in a pie chart, just to name a few. Our development is close to that approach but for the edX platform instead of Khan Academy. The visualizations used the Google Charts library.

A proposal on how to measure and infer a set of high level indicators from low level Khan Academy data [37] was also presented. This high level information included total use, efficient use, activity time distribution and gamification and exercise-making habits. The parameters were tested in a real course at Universidad Carlos III de Madrid and allowed to draw conclusions relevant from the learning process based on this case study.

In September 2013 a learning analytics module that extended the native support of the Google Course Builder (GCB) platform [31] was implemented. It incorporated functionality to capture and store YouTube events, visualizations of student performance and to assist teachers in informing them on possible course materials susceptible to improvement by making automatic recommendations in this sense.

Most of the tools we have seen so far provide ways to process and present data from the logs. We would now like to mention another type of analytics tools using the logs to make forecasts. For example, in 2014, A4Learning (Alumni Alike Activity Awareness) [26] was presented. A system destined to provide students with an estimation of what will be their grade based on how they are currently doing and comparing it to previous course editions. The goal is to increase student's self-awareness of whether they are falling behind or not.

In October 2014, a generic methodology for determining students effectiveness in the use of virtual learning resources was proposed. It targeted MOOCs and SPOCs and established the steps to define precise metrics in this context. The method was

called PES, standing for "Precise Effectiveness Strategy" [36]. PES comprises four phases enumerated below:

1. Selection of the educational resources and activities.
2. Calculation of the effectiveness for each individual resource and activity.
3. Calculation of the effectiveness for all the resources and activities of the same type.
4. Calculation of the global effectiveness of the course.

PES guidelines recommended to consider connections or relationships among the resources, the nature of each of them and a continuous scale to measure effectiveness.

2.3.1 Learning analytics studies using edX data

Outside the edX platform but taking data from its logs studies have been carried out to better understand student behaviour and the learning process. One of these studies [25] has been centered in getting insights from video dropouts and interaction peaks in MOOC videos. It claims to be the first study of its kind. The team has found that there is a direct relationship between the video length and the students stopping to watch the video. It also uses its video prototype analysis tool to detect peaks and sinks in the video playing. Further analysis on the video content or transition at the time of this standing out events, enabled them to draw conclusions and make recommendation for video authoring and interface. The analysis was done using data for four courses offered in Fall 2012 comprising hundreds of videos and tens of thousands of students. Our project provides a visualization (Repetitions per video intervals) for the video watching profile allowing course staff teams to detect these peaks and sinks on a "coursewise" level.

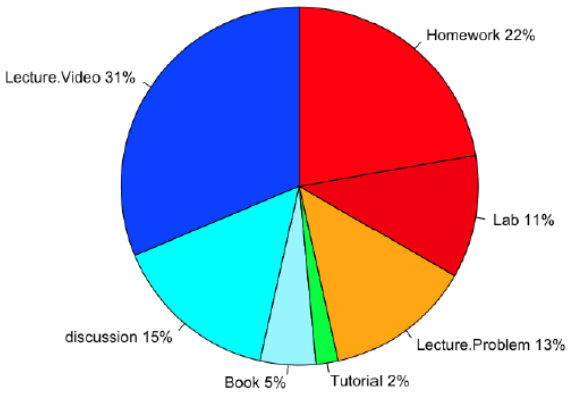

Figure 2.4: Reproduction of the student activity distribution for study [6].

A team of six researchers from the MIT [6] studied the correlation of skill and improvement with a students time on tasks using log data from two MIT-offered

MOOCs: Circuits and Electronics (6.002x) and Mechanics Review (8.MReV). They built a pie chart reproduced in figure 2.4 to determine student time distribution in the course instructional resources: lecture videos, discussion forums, eTexts or books, tutorial, lecture problems, laboratories and homework. Together with statistical tools, the chart helped them to "find strong negative correlations [...] between student skill and resource use", which they explain based on the initial skills of the students taking the subject. They also found "weak [...] correlations between relative improvement and resource use".

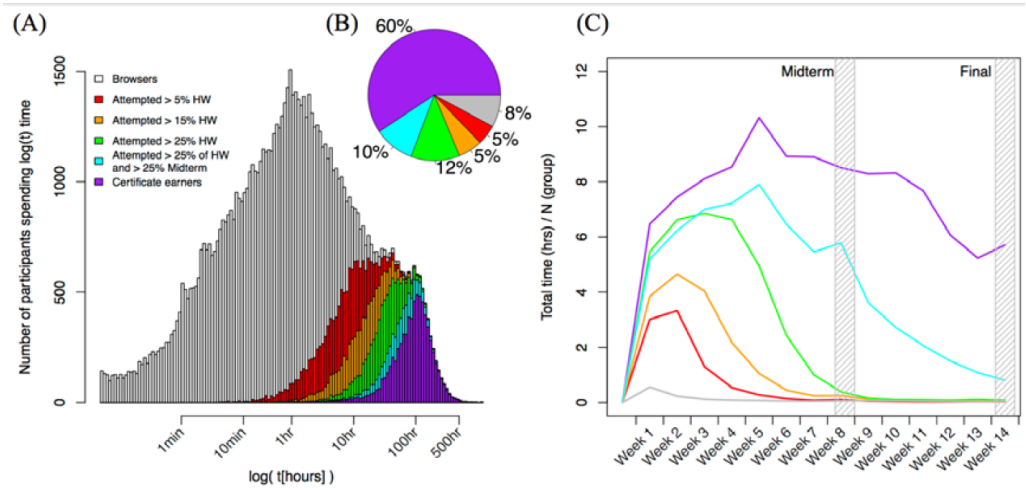

Figure 2.5: Reproduction from [46]. "(A) Distribution of time spent by 108k participants in the course ($\sim$ 7200 received certificates). We have divided the non-certificate earners into tranches (colors) based on the percentage of assessment items they attempted. (B) Percentage of total measured time in the course, and (C) average time per student per week for each tranche."

A previous study of another MIT team [46] with data from the first MITx course, Circuits and Electronics (6.002x), used in the above-mentioned study together with another course, showed the importance of measuring the time allocated to courses by students in determining which of them were more likely to finish the course successfully. That is, "distinguish browsers from certificate-earners". A reproduction of some visualizations used in that analysis appears in figure 2.5. The Gaussian-like curves in (A) show the huge amount of explorers (area under the curve), that spent in average an hour in the course. Students that have attempted to do more than a 5% of graded assessments have allocated a time that ranges from ten to a hundred hours in average. Those who earned a certificate appear at the rightmost area of the chart accounting for the largest number of hours and the smallest number of students. This visualization does show, at least for that specific course, that there was a clear relationship between time-spent and passing the course. In (B) is obvious that in spite of passing-students being a minority with respect to the number

of total enrolles, they dominate the total time all learners spent in the course by a sound 60%. Their behaviour may be characterized in terms of effort, dedication and discipline. They perseverance and more steady dedication is better described by the total hours spent on a weekly basis in part (C) of figure 2.5. The behaviour of those who did not passed the course is characterized by the red, yellow, green and blue curves that resemble a Rayleigh distribution deflated by the middle of the course except for those who attempted more than a quarter of the total assessment tasks. The latter seemed to be struggling or at least interested until the end, but their dedication did not translated into success. This is the group more likely to join the certificate earners. A teacher may decide to focus on this group to see what they did not make-it in spite of being interested or active up until the course end date.

The authors expressed their believe in the potential of learning analytics to "provide useful information to teachers, to resource creators (authors), and to members of organizations trying to improve their MOOCs".

2.4 Learning analytics in edX

Basic analytics are currently on display at the platform. For students, a basic Progress page is the place where they can track their evolution during the course. This page consists of two parts: a progress graph and the same information in text and number format. Progress here is understood as graded problems completed. Problems are considered those contents where grading makes sense such as homework, laboratories, essays, questions and so on. No grading policy is applied. Figure 2.6 illustrates this progress graph with an example. The percentages of graded problems completed per chapter are represented in a bar graph. Interaction is added so that when the mouse is passed over columns, a little more detail of that score appears. "Ex 01", "Ex 02" and "Ex 03" in the horizontal axis refer to the three chapters, namely *Example Week 1*, *Example Week 2* and *Example Week 3* respectively, of edX's Demo course, from which the chart is taken.

As this is a chapter aggregation score, below the graph, the page on edX shows the student the detailed scores per modules where score apply. Scores are given in sequential order, no name or description provided, just the order they follow in the chapter. Below this information, the student is provided with scores about exams and certificates. Instructors have their own tab/page with a series of subsections like *Grades*, *Admin*, *Forum Admin*, *Enrollment*, *Datadump*, *Manage Groups*, *Email* and *Analytics*. The Analytics section reads: "No Analytics are available at this time." The Datadump section enables the instructor to export to a CSV file all student profile data or all the responses to a certain problem. The instructor could then process that information if interested outside the platform. Instructor is offered with a "Course statistics at a glance subsection" where apart from course technical

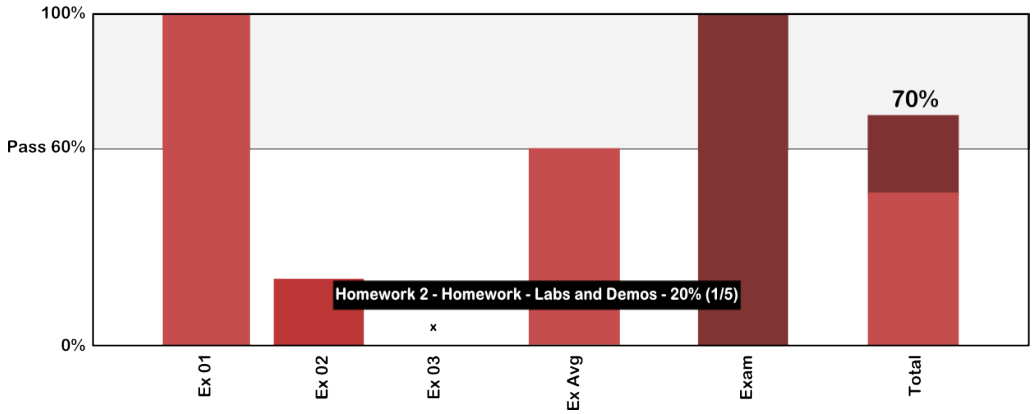

Figure 2.6: EdX course progress chart.

information as number of descriptors for module type he can only get the number of enrolled students. A beta dashboard version they are working on would show instructors the following student distributions: per-problem scores, year of birth, gender and level of education.

On September 30, 2014 edX started to release new visualizations on Insights[1] in three categories: 1) Enrollment Activity, 2) Enrollment Geography and 3) Engagement with Course Content.

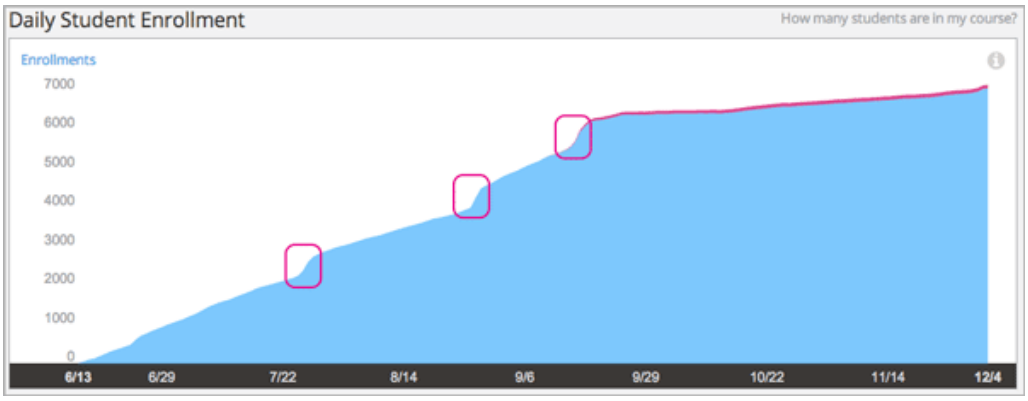

Figure 2.7: Example of daily enrollment evolution for a MOOC [62].

The enrollment activity visualization is of stacked area type, meaning that the area under the curve accounts for the number of registered students. It includes the interactivity of showing a box with added details when mousing over the curve, containing a break down of enrollment per type. An example from the documentation is reproduced in figure 2.7. A period of almost linear increase rate is visualized for a number of weeks, except for three particular time slots highlighted in the chart that show an accelerated increase perhaps due to particular events such as publicity.

[1] http://edx-insights.readthedocs.org/en/latest/change_log.html

Course teachers would be the ones having a semantic hypothesis for the increment. It comes the time when a marginal number of students still continue to register but at a very slow pace, marked by an almost horizontal trace.

That visualization was for a MOOC. In contraposition, figure 2.8 displays another example of the visualization for a private course taught online. It is no surprise here that registration pattern resemble a uniform distribution. At course start most of the students enroll for the course and the number keeps constant perhaps due to the the fact that enrollment process was closed. In this example the annotation box with the number of students per registration option is also displayed for a specific date.

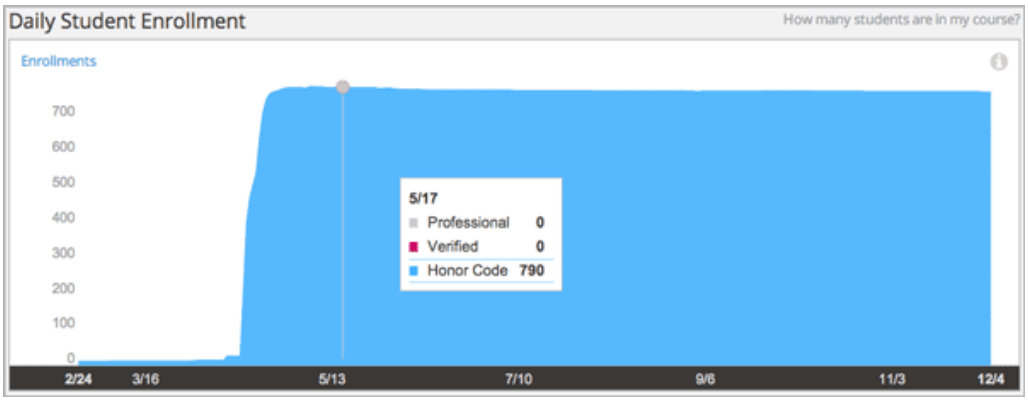

Figure 2.8: Example of daily enrollment evolution for a private online course [62].

It is a well-known practice when delivering speeches or making presentations that you must know your audience and customize the message for better results. It works in an analogous way for teachers. If the course content is tailored to the majority of the students, the outcome is likely to be more satisfying for both students and teachers themselves. This is where geographical information plays its role. Insights incorporates a mixture between a map chart and a heat map to present the geography of course enrollment in a visualization like the example of figure 2.9. In a world map the density of the course students is indicated by means of a color scale from cold to heat tones of blue in the example, where heat tones express higher concentration as usual in this type of charts. The graph uses also a mouse-over event to display country name, number of enrolled students from that country and the percent they represent of the total enrolles, depending on cursor position. This way, in a quick glance teachers are presented with an eagle viewpoint of where in the world learners are following their courses.

The third visualization category in Insights is related to student activity or engagement with course content. These are line charts with three lines representing the number of active students, how many of them have watched a video and how many of them have attempted a problem. A student is considered active when he/she has

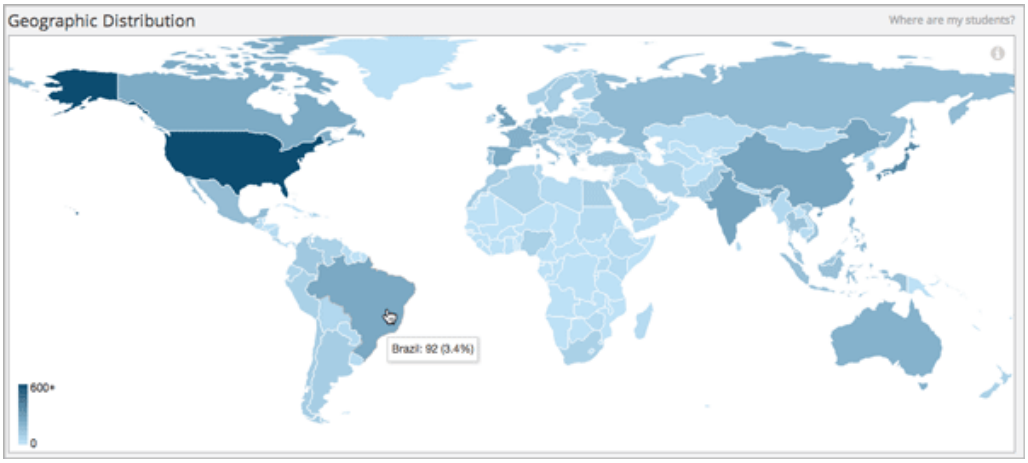

Figure 2.9: Student's geographic distribution chart [62].

logged in to the platform in the past week at least one time. An example of this visualization for a Small Private Online Course (SPOC) is reproduced in figure 2.10. The example reads that students for that course were more prone to try a problem than to view a video. It could be interesting for a teacher to correlate the peaks or abrupt changes with course events, deadlines or assignments to draw conclusions.

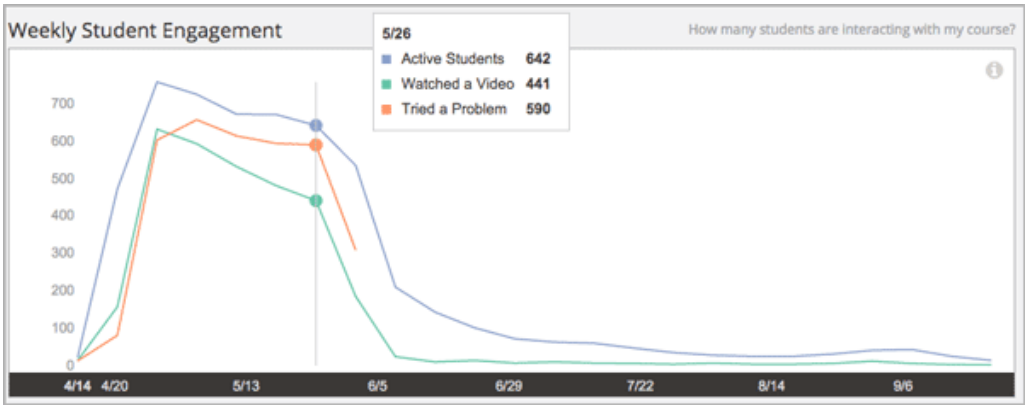

Figure 2.10: Student engagement visualization example for a SPOC [62].

An example of these correlations appear in figure 2.11 for a MOOC, highlighting course start and first homework due among other two events. The peak clearly correlates to the first deadline. In this case, there were more students watching video than solving problems. It may depend also on the nature of that first homework in the sense of what was there asked to do.

Notice that apart from the progress graph, the remaining of the edX visualizations are exclusively for teachers and course staff. Here, opportunities are identified to improve the information that can be given to student about their own learning

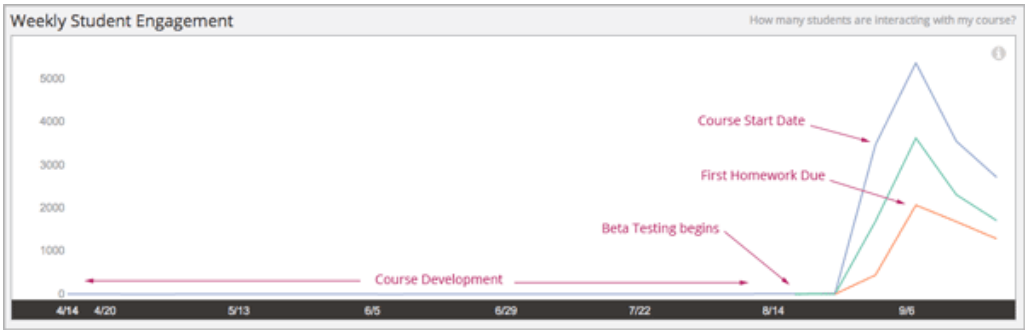

Figure 2.11: Student engagement visualization for a MOOC [62].

process. Learning analytics is as important for students as it is for teachers. They are respectively individual and collective decision-makers, and the decisions they all make impact the learning outcomes.

2.4.1 EdX further initiatives on learning analytics

The edX team is aware of the need to transform the available volume of raw data into information relevant to students and instructors, that could be extensible as well to educational researchers. The most visible effort in this sense have been the projects *edX Analytics Service*[2] and *Insights*[3], a Python+Mongo+Django framework for creating simple, pluggable analytics based on streaming events, according to its own homonymous GitHub repository definition. It is important to keep in mind these are both projects yet to be finished. *Insights* is thought to work outside of edX getting the information by reading database replicas. For that, it relies on two tools specifically written for edX: *loghandlersplus* and *djeventstream*. In this order, a log handler able to stream the events out and a framework for receiving events as JSON dictionaries. Once working the event flow, Insights is meant to be an API for handling those events. The Insights user could also query the LMS system for data replicas and define views so that this data is represented in dashboards in the desired graphical way. Being a Django project, the views need to be coded in Python which is a clear limitation to students as well as instructors, who do not necessarily have programming skills. For the vast majority of students and instructors to get useful learning analytics by coding Python using Insights is thus not an option.

The Insights project, once part of the Open edX Platform, has been removed from this list of current open projects related to edX that encourages developers to contribute. Perhaps, this can be interpreted as a deprecation or project discontinuation for reasons unknown to these authors. Recently, the edX team have been working in the edX Analytics API Client[4], a tool that allows users to retrieve data

[2]https://github.com/edx/edxanalytics
[3]https://github.com/edx/insights
[4]https://github.com/edx/edx-analytics-api-client

from the edX data warehouse.

Regardless of continued efforts to support edX with the learning analytics the platform deserves, this kind of info is still missing. Some developers who have forked the main repository into their own ones, are contributing with their initiatives to improve this learning analytics context.

Outside the platform, however, an Open edX Analytics Home[5] page contains updated info on Data Packages distribution, aware of the valuable information edX generates, while waiting for comprehensive learning analytics built into the platform.

Here the necessity of a learning analytics module or extension of edX for the stakeholders (mainly students and teachers) to take advantage of the high volume of low level, raw data available. Making the learning process more meaningful will make it also more rewarding.

2.5 Visualizations

The Khan Academy module has powerful visualizations. Its "Progress" navigation tab is divided into four sub-navigation items: Skills, Videos, Activity and Focus. By default, Khan Academy displays the most recent days but the range is customizable for the user to examine his/her progress in the time interval they desire. The content of the menu entry "Skills" is displayed in the screenshot of figure 2.12. It presents the area of knowledge in which the user is currently learning. In Khan Academy they call it "the Mission". A bar shows the progress in percentage over the total number of problems that the mission involves. Khan Academy identifies problems with skills as students show their ability by solving those. A check mark allows the user to display only the skills attempted in the detailed categories below. "Missions" are divided in categories and this categories in turn are broken down into skills. Categories can be collapsed to facilitate navigation according to the user interests. Skills are listed in table format with the level acquired with them, the number of questions and the time in minutes spent working in each skill.

The second progress sub-navigation tab is related to Videos. Videos are the equivalent to lectures. Therefore, video watching can be assimilated to class attendance. Videos are grouped per days on which they have been watched. The full list appears on a daily basis and at the rightmost column the total time of video watched that day is shown. A capture of this screen is reproduced in figure 2.13.

An activity visualization is the third progress sub-tab as shown in figure 2.14. It is a stacked column chart accounting together minutes on video and skills time on a daily basis. The chart has a dual vertical axis, one for time and the other for the energy points earned, the gamification motivation of Khan Academy for their students. The chart is also interactive and offers detailed information by mousing

[5] https://edx-wiki.atlassian.net/wiki/display/OA

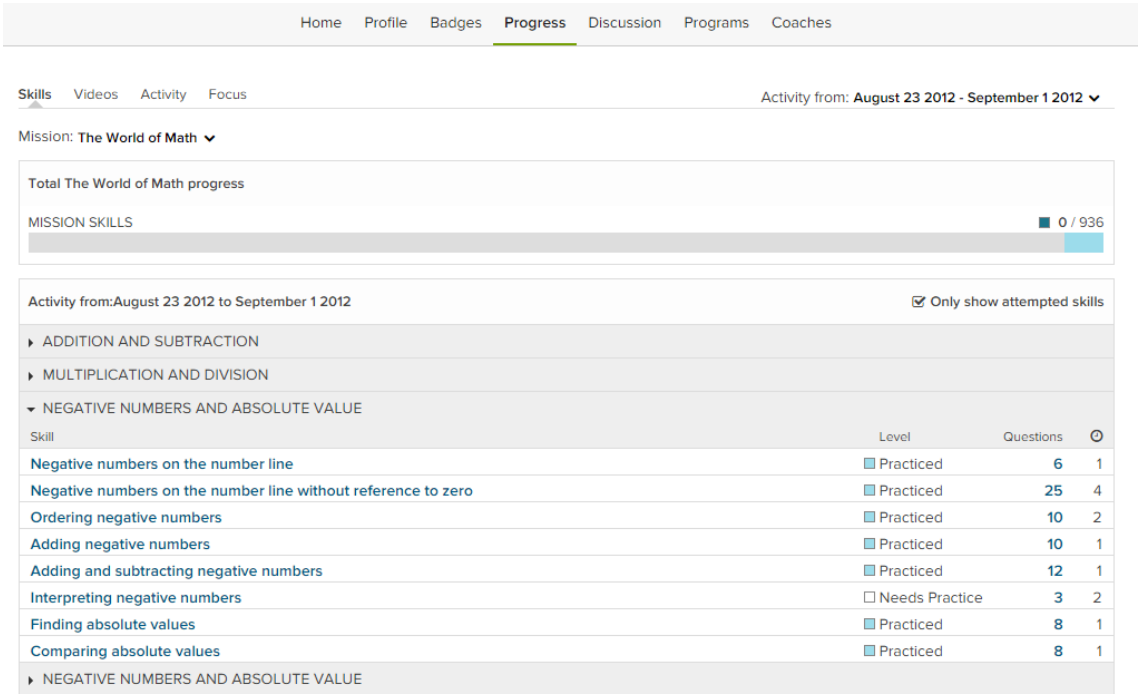

Figure 2.12: Skills summary in Khan Academy.

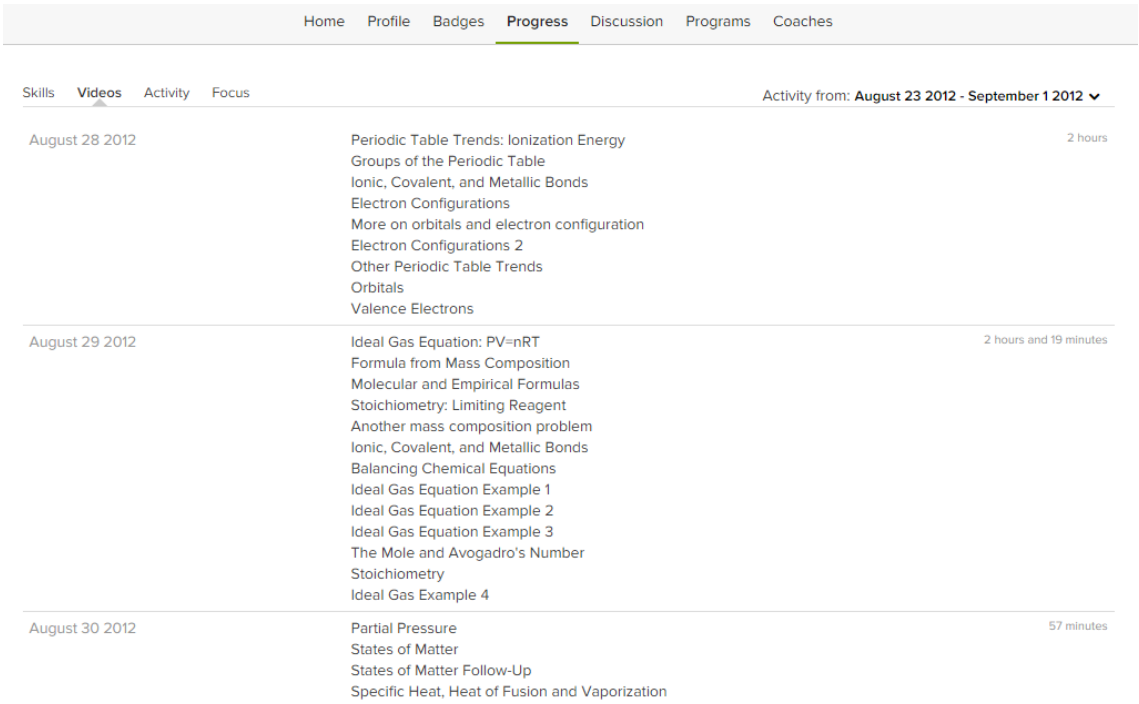

Figure 2.13: Video summary in Khan Academy.

over the individual days. In the figure, details for a single day are shown including energy points earned, time on videos and list of those videos (the main ones if they are many) and the badges earned to reward accomplishments.

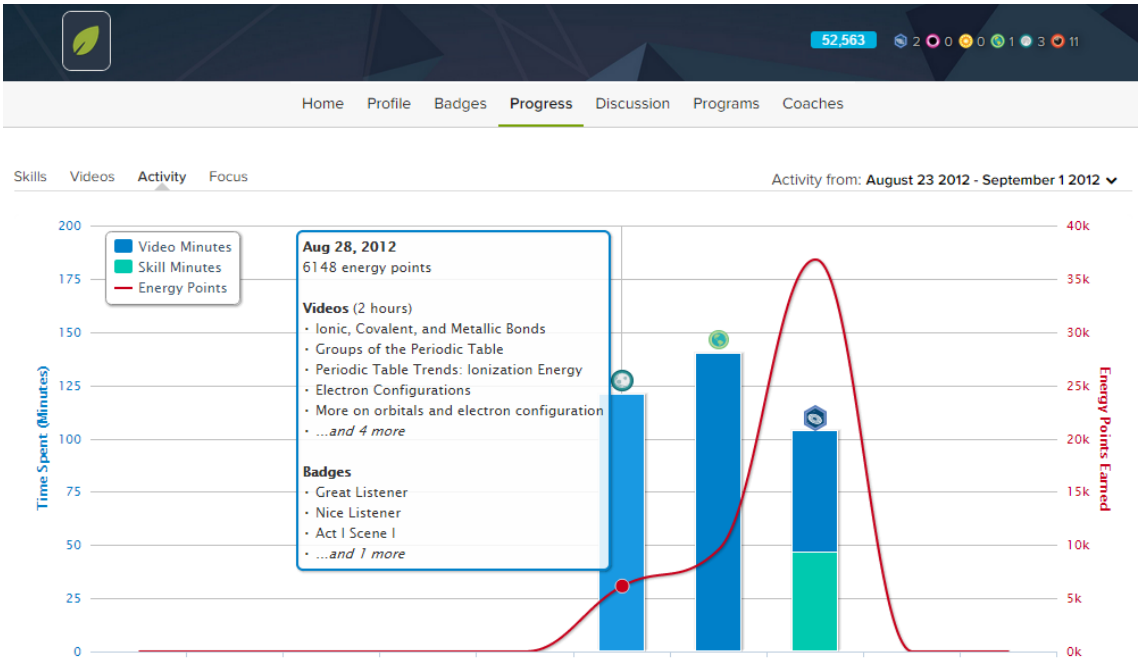

Figure 2.14: Activity visualization in Khan Academy.

The fourth and last progress sub-navigation tab is what they call the "Focus" visualization. As it can be seen in figure 2.15, it consists of two concentric donut charts, the inner for videos and the outer for skills. Each shows for the selected range of dates the percent of time allocated to every video and skill task respectively. The interaction with the visualization provides more information by mousing over the chart slices.

Apart from Khan Academy, in March 2013, a set of visualizations about learners' emotions in a computer interaction-based environment [28] was presented. The importance of this approach lies in that learning is above all an emotional process. Models linking events to the appraisal of emotions were used. A commonality among these models is goal achievement categorization as favorable or unfavorable. Particularly, the Hidden Markov Models part from a set of possible observations and a set of states. For the latter five states were defined: 1) working on task, 2) finding a problem, 3) looking for solution, 4) solving a problem, and 5) being distracted. For the observations the possible students actions in the environments were taken, e.g. code compilation free of errors, browsing course resources, browsing the web, editing code in the text editor and so on. Four emotions were the target of the visualizations: happiness, frustration, confusion and boredom.

Figure 2.15: Focus visualization in Khan Academy.

Chapter 3

Requirements

In agreement with the declared objectives, this requirements section describes in detail the application to be designed and the visualizations to be implemented and incorporated to an analytics section on the navigation menu of the courses at edX.

The use case UML diagram for the application is shown in figure 3.1. There are three actors: students, teachers and the server. Students can visualize only their own data for the sake of privacy policy, but to do that they will have to log in to the platform first. For the visualizations displaying information for a single video, they will also be able to switch the video whose data is being visualized. Prior to data visualization, the application has to request data to the server.

Teachers on their side should have a selector to chose a single student, i.e. any of the course enrollees, or among the specific aggregates defined for the visualizations if any, typically the class average. That way, teachers are enabled to access the whole picture as well as at the individual level. Teachers can do anything a student can, as indicated by the generalization relationship.

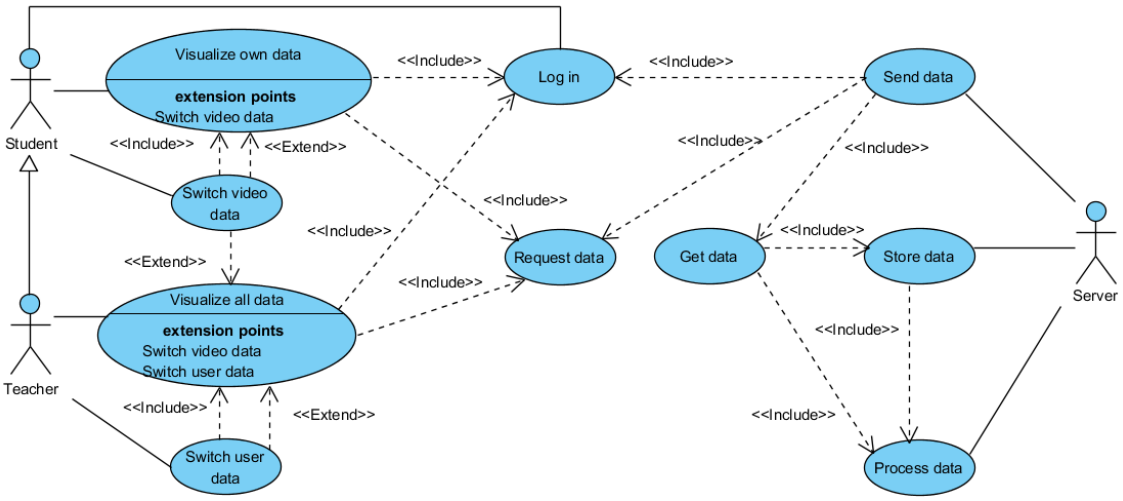

Figure 3.1: Use case diagram for the application.

Servers, the non-human actors in the diagram will process, store and send data requested from students or teachers. To store data they first need to process it. To send data the server will ensure the user who requested the is logged in to the platform. For sending data the server gets it provided that it was processed and stored before. In addition, data will only be sent if a request is received.

This is the high level use case of the application. Now, before examining the requirements for the visualizations to be implemented, we discuss why we have chosen those visualizations.

EdX have a single progress graph for students. That visualization shows only graded information, that is, the scores the student got from homework assignments, laboratories, partial and final exam, together with the total mark for the course so far with a pass threshold on 60%. Usually this is the information most relevant for students, especially for those opting to get credit for the course. This is closely related, if not the same, to traditional courses, where at a minimum, the feedback students get from their teachers are their marks. However, there is a potential in MOOCs for automatic feedback to be provided on time management and resource consumption, in addition to the primary grading information. To know how they spend time on the MOOC, to what extent they consume which type of resource and what was their a posteriori learning schedule could benefit students in giving them an insight into how they learn. We all have a feeling or intuition on what is our learning method or learning style, what our learning patterns are, what works for us and what not. In the MOOC era, that intuition could get confirmed or backed by data, objective information, or proved to be a subjective, unreal feeling. This is not only a picture of the past but a tool students could also use to plan and optimize how to tackle the next MOOC, or even the next learning adventure they get involved in.

And what about teachers? Much like for students scheduling is an extra feature, for teachers is part of their job. Time distribution on tasks and resources is a powerful indicator for them to make a data-driven course scheduling assigning time slots proportional to the time constraints inherent to the course. Especially useful as feedback for the following runs of the course.

A third collective that could also profit are researchers. These visualizations provide them with another tool to corroborate or discard hypothesis.

3.1 Visualization: Video time watched

This visualization should display two percentage indicators for every video in the course. The first percentage would be that of video length the student have watched. No video fragment should be accounted for twice. The idea is, a student can view for example five times the first twenty seconds of a one-minute video. That means he has watched the video for an amount of time equal to video duration. However, that does not mean they have visualized the whole video. The second percentage

of the graph, to appear by the first one, is the total time spent by the student on every video, relative to its duration.

We will nominate the first measure *non-overlapped percentage* and the second as *total vs. video length*. No surprise with the names. They will be both values to appear in the dimension of the y-axis, while on the x-axis the names of the videos will be shown in the order they come in the course. A legend with the indicators will also be displayed.

Non-overlapped percentage is bounded to 100% by definition, which is by the way the value that will make teachers the happiest since it means that at least the video was fully played. Note this is the most we can assert. The fact that video is played from start to end does not imply necessarily that the student viewed it all, nor even a single second, but it is the best we can do. The scenarios are multiple and range from accidental to on purpose. An example of accidental: the student receives a phone call when he has just clicked the play button. They could also press play and do anything other than watching the video. This is very important in that we need to be very careful when interpreting results.

On the contrary, the total relative time on video has no boundaries. The most time a student spend on a certain video, the larger its value.

The aggregated indicator that should be available to teachers for this visualization is the statistic mean of all the students taking the course.

3.2 Visualization: Video time distribution

The purpose of this visualization is to show how students allocated the whole time they devoted to video, within the individual videos in the course. This is another way of visualizing the total time in the previous visualization. Of all the time spend on videos, what was the percentage for every video. Unwatched videos of course are not included as their percentage is null. As it all makes a hundred percent, a pie chart would be the natural choice. No weighting policy is to be apply. Therefore, if a student has viewed completely all of the videos without repeating any segment, the video with a longer duration will appear in the visualization with the largest slice. This must be taken into account when interpreting the visualization.

For the teacher it is available the enrollees' aggregate where all their individual video times are added up.

A legend for the visualization will show the name of the videos associated with each percentage.

3.3 Visualization: Problem time distribution

This is the version for problems of the video time distribution visualization. Here the potential to drive scheduling is even bigger since videos have a duration that could be used as a reference but that is not the case for problems. Problem time

will be computed as the time the student spends on it, starting when the problem is displayed to him/her and ending when he/she moves to another section or resource within the courseware. Note the latter point is not that when student submits the problem solution for correction. It takes into account they could keep reflecting on the problem after they have made the submission. Several attempts are often allowed and students could change their mind or the solution they first chose or proposed. Even if that does not occur, all the time the problem is on screen at the course is accounting for this indicator.

This way in which time on problems is measured cannot be forgotten at the time of reading the visualization information. The learner could have finished the problem and submitted the answer getting then away from the device from which he is accessing the course for any reason. This might lead to a huge inflation of the time the student has actually spent on the problem. Therefore, a threshold should be established to limit this error. Here that threshold will be of thirty minutes. We must acknowledge that it is neither optimum nor fair to impose the same threshold to all the problems as their difficulty level varies as well as the skills they require. Let us consider different question styles available at edX according to the *Demo* course aimed at presenting the platform and the educational features it has to offer:

- Pointing on a picture.
- Drag and drop.
- Multiple Choice Questions.
- Mathematical expressions.
- Chemical equations.
- Numerical input.
- Text input.

It is clear from the above list how unfair it is to bound the maximum time on them with the same threshold. It is however the best we can do. The desirable way for problem-threshold customization would be perhaps creating for it a meta-data attribute in the database entry for the problem. The model could be a default value that would be modified if the course author (typically the teacher) sets its value, as they are who could make the best based-on-experience estimation of the time most suitable for the threshold. Since this modification goes beyond the scope of this project the threshold is unfairly the same for all problems.

Problem names will be displayed in a legend related to the percentages on the visualization.

3.4 Visualization: Repetitions per video intervals

This visualization should show the number of times every video interval was played. There is no fix video interval. This is determined by the intersection of the different (if any) video reproductions of a single student or the whole of them. To clarify it

with an example, let us suppose a first student watches a thirty-second video from second 0 to 5, a second student views it from second 10 to 18 and a third student plays it from start up to 17th second. The interval division and the number of times each was played is as shown in table 3.1.

Interval (seconds)	Times	Student(s)
0-5	2	First, Third
5-10	1	Third
10-17	1	Third
17-18	1	Second
18-30	0	None

Table 3.1: Basic, dummy example for video-interval division and repeated times.

The abscissa axis of the chart will comprise from zero to video duration in seconds, meaning that its range will differ among videos as will video-length. The determined intervals will have an horizontal bar whose height will be the number of times it has been played. The ordinate values will then always be integers.

The visualization should be accompanied by a selector to chose the video among those of the course for which to show the repetitions. The video list will come sorted by order of appearance throughout the course. For teachers, another selector is to be implemented to be enable to view the individual student information together with two aggregates. On the one hand, the overall repetitions profile, that is, including all the times every single student has watched the intervals. On the other hand, the repetition profile where each student is accounted for one time at the most. This second aggregate removes the impact of an individual learner responsible for a high number of repetitions.

This visualization is not only intended to display the student's consumption distribution for a particular video, but also to serve as a tool to assess the quality of video content. The representation stand out high-repetition intervals. And why is that important? Because it could prove valuable to detect those intervals especially difficult or on the other side, widely popular, among students. Of course further assessment is needed to check if the popularity lies on the dark or on the bright side, as the visualization simply makes the most repeated intervals to stand out. Video segment popularity on the bright side, means that it is a good practice to try to extrapolate that segment to future videos. That goes through identifying the specificities that made learners love that fragment. The darker popularity means there was something in that video interval that remained unclear to students: a concept, an idea, an explanation, a solution, something. As a result, they had to play it over and over to try to understand it. In further runs of the course, or of another course reusing that video, the latter could be improved in those segments students found particularly difficult, trying to clarify its content and using alternative ways to present it.

This feature of video assessment is of especial interest for course authors and content creators, who are often the same person but they do not have to.

No legend is required for this visualization.

3.5 Visualization: Daily time on video and problems

On the horizontal axis of this visualization there will be dates indicating year, month and day. The vertical axis should be a time axis. For every day a problem has been attempted and/or a video has been played there will be an entry on the x-axis, with their corresponding time allocation on video and problems side by side. Videos will not be individually distinguished by name or any other label or attribute, just video time in general. All the time spent on videos will be added up. The same for problems. A legend should be visible to tell between the video and the problem indicators.

A selector for teachers will enable them to chose among single students or their average computed as the statistic mean.

This visualization helps assess student dedication to course and time-effort. Is their effort steady or has a peak prior to exam or at some specific points? It is one of the questions that could be answered.

The graph is more powerful when correlated with course schedule to check how learners reacted to the topics and resources as they were released.

3.6 Visualization: Video events distribution within video length

In this last visualization, a video interaction profile is to be presented. On a horizontal line for each video event, the position where the event took place throughout video length will be shown.

The events to be considered are:

- play,
- pause,
- change speed,
- seek to position, and
- seek from position.

The horizontal axis will be similar to that of the repetitions per video intervals visualization, spanning from zero to video duration in seconds. The vertical axis value is meaningless, being only required that occurrences of the same event are at the same height.

A legend is to be included to distinguish among events.

Two selectors should be incorporated, one for videos in course-appearance order and the other, only seen by teachers, to select the student or the aggregate of them. Here, in the aggregation, no processing makes sense. It is just the union of the events from all learners.

This visualization is like a radiography of student interactions with video.

Chapter 4

Design and implementation

In this chapter all the steps taken to create the visualizations detailed in the requirements will be described from scratch.

4.1 EdX Developer Stack

The developer stack, shorted devstack, is the edX portable development environment using Vagrant so that it can be reproducible, thus facilitating and encouraging community contributions. "Vagrant is a tool for building complete development environments"[1] that lowers setup time, reads its web page.

It makes no sense to try to explain here a step by step devstack installation since it is not the purpose (yet it was indispensable of course to carry it out) and it is at risk of becoming obsolete or useless by the time this report is published.

However, a quick explanation will put us in context. It all starts by installing VirtualBox and Vagrant in that order. Both of them are straightforward. The Vagrantfile, a Ruby-syntax document, which contains configuration and provisioning information for the virtual machine, is downloaded. Here ends the straightforward world. Good news is the error you found is very likely to have been addressed in forums. Next step is to install the VirtualBox Guest Additions plug-in for Vagrant. Now it is time to bring the virtual machine up with the command below (the dollar sign symbolizes the terminal):

$ vagrant up

For the most recent installation instructions check [59].

The first time the base box is downloaded and the next times reused unless it is destroyed and recreated. While the machine is booting this information for external access to the virtual machine via SSH appears on the terminal:

SSH address: 127.0.0.1:2222

SSH username: vagrant

Then shared folders are mounted and with NFS technology access is possible both from host and virtual systems.

[1] https://www.vagrantup.com/about.html

To next three commands are used to enter the machine, log in as *edxapp* and start the Django development server respectively:

$ vagrant ssh

$ sudo su edxapp

$ ./manage.py lms runserver –settings=xinsider 0.0.0.0:8000

Figure 4.1 is a screenshoot of the terminal when the three of them are executed. It illustrates the welcoming and warning message of the machine and at the bottom the URL where the server is running. The server does not have production capabilities since as the Django staff says, they are "in the business of making Web frameworks, not Web servers"[2]. However, it lacks nothing we need for development.

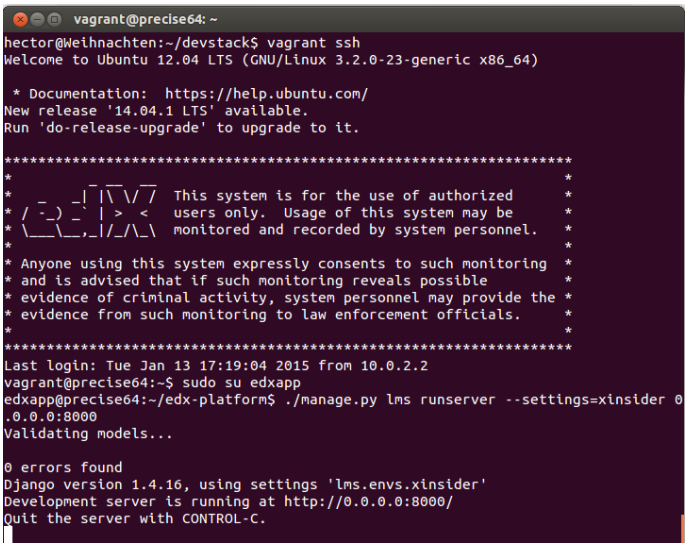

Figure 4.1: Screenshot from the devstack virtual machine on the terminal.

It is strongly recommended to stop properly the vagrant machine as every development session finishes, in order to avoid the error below.

pymongo.errors.ConnectionFailure: could not connect to localhost:27017: [Errno 111] Connection refused

The correct way would be as shown in the screenshot from terminal in figure 4.2. Provided that the server is running and the Terminal window is active, Ctrl+C is pressed to stop the server. Then the command $ exit is typed twice to log out as user *edxapp* and to the machine respectively. Now, the command $ vagrant halt will shut down the machine.

If it is too late for prevention, to correct the error the following commands should be typed when logged in to the machine:

```
sudo start mongodb
```

[2]https://docs.djangoproject.com/en/1.4/intro/tutorial01/

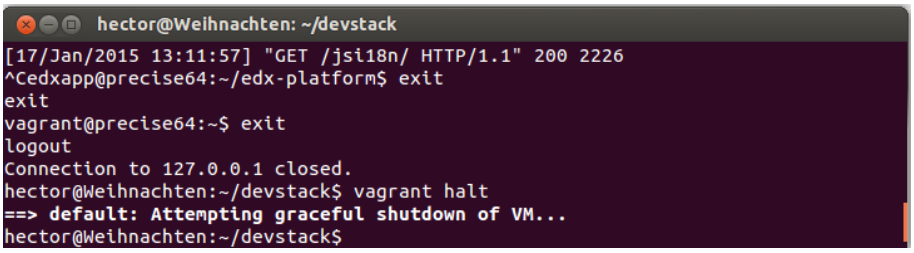

Figure 4.2: Screenshot from the terminal to show proper machine shut-down.

This are the basics to start and stop the machine and the development server.

Devstack comes with the *DemoX edX Demonstration Course* preloaded and the four dummy accounts listed in table 4.1 for test. Their names are representative of the user kind they represent. Thus, *staff* refers to a course staff member, typically a teacher or team of them. This account enable the developer to see the dashboards and information tailored for teachers and is useful to test platform additions oriented to this role. Users *honor* and *verify* symbolize the two homonym enrollment options, meaning that by successfully passing the course an honor or verified certificate is to be granted. Honor certificates is just a kind of souvenir for the personal satisfaction of its owner. Verified certificates serve the purpose of getting credit for the course with a document legally recognized and granted by the participant organization in charge of the course. This is one of the most beloved features of edX and research has shown that certificate earning thrives completion rates and act as a powerful motivation factor.

User name	E-mail	Password
audit	audit@example.com	edx
honor	honor@example.com	edx
staff	staff@example.com	edx
verified	verified@example.com	edx

Table 4.1: Preloaded dummy accounts in edX devstack.

4.2 Tables in edX databases

EdX uses both relational and non-relational database systems. MySQL for the former and MongoDB for the latter.

The *modulestore* Mongo collection stores course structure and content from which the courseware is rendered. The MongoDB collection follows no particular order. Course structure is stored using a top-down tree approach where every node knows about its children if any. The logical underlying hierarchy is represented in the

scheme of figure 4.3. Courses are the root elements, the outer containers. The next hierarchical containers are chapters, sequential (or video-sequence) elements and vertical (or problem-set) elements. Course authors decide on wether to use sequential or video-sequence depending on the semantics of the group of elements targeted. Video-sequence does not necessarily have to contain only videos, but it used when these kind of resources predominate or is the gist of the group. In an analogous way, vertical or problem-set containers are used. These pairs are containers of the same level and the difference between them is only semantical.

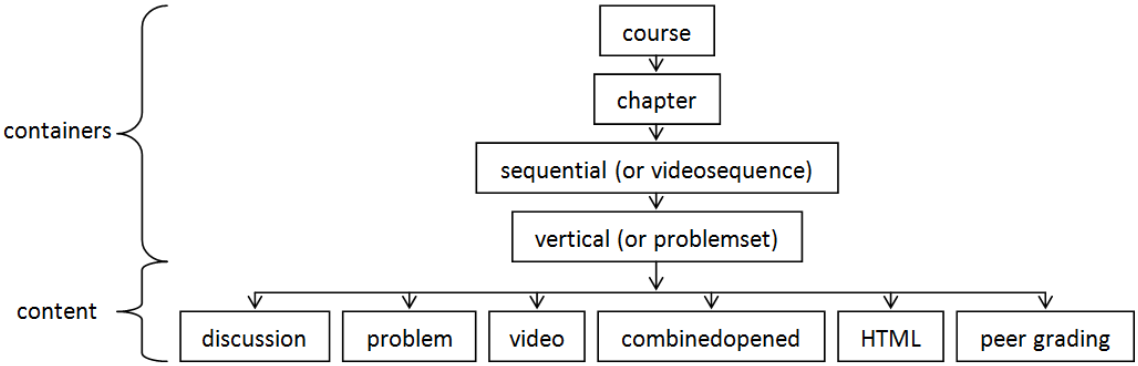

Figure 4.3: EdX course structure.

Content is located at the leaves of the tree. It includes, but is not limited to, HTML, video, problem, discussion, combined-opened and peer-grading.

Once the desired course is accessed, courseware navigation looks like showcased in figure 4.4, on which the underlying containers and content elements are drawn, and labelled according to the database nomenclature. This superposition is very illustrative of the relationship among collections stored in the non-relational database and how they are rendered to be presented to the platform users.

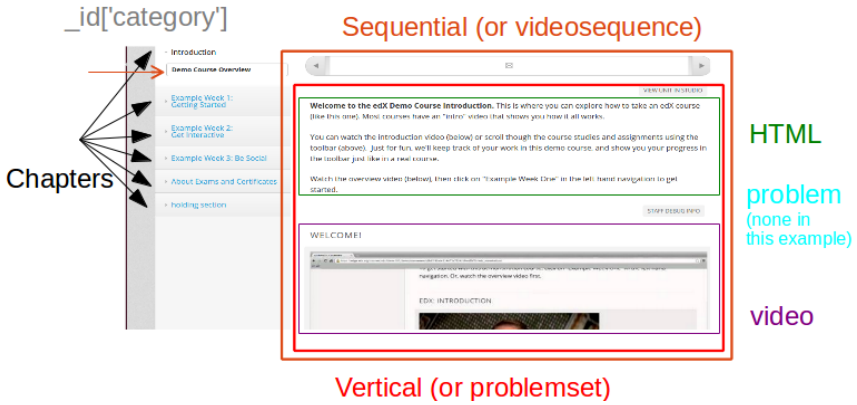

Figure 4.4: EdX course structure drawn over screenshot: containers and content.

On the other side, the MySQL tables save user information. A few of the latter, potentially relevant for this project, are listed in 4.2. There, strings are encoded using UTF-8 and dates are stored in UTC (Coordinated Universal Time) for convention.

The value of *id* field in *auth_user* table is the same as that in the MongoDB database's users collection.

The demographic data in table *auth_userprofile* is that entered in the registration fields. Every row in this table corresponds to one row in *auth_user*.

In table *student_courseenrollment*, a row is created for every student who starts the enrollment process, even if they never complete registration.

Metadata in table *user_api_usercoursetag* uses the key-value pair format. An example could be the partition and group of a student in a course with randomized groups for A/B testing.

Table *courseware_studentmodule* is very valuable for learning analytics. There is a separate row for every module a student accesses. Therefore, this is the largest default-enabled table[3].

Information relative to obtain credit for the course is stored in the table *certificates_generatedcertificate*. It includes the state of certificates and final grades for a course.

Table name	Description
auth_user	User login and permissions.
auth_userprofile	User demographic data.
student_courseenrollment	A student's enrollment for a particular course run.
user_api_usercoursetag	Metadata for students involvement in courses.
courseware_studentmodule	State and score for courseware modules.
certificates_generatedcertificate	Course certificates data.
track_trackinglogs	Event logs.

Table 4.2: Selection of MySQL tables at edX.

The table for the *TrackingLog* model is disabled by default presumably due to its potentially rapid-growing size. There is a setting option for limiting the maximum size of this table. Nonetheless, event data is written primarily to a JSON-format, log file that is periodically delivered to data czars of partnering institutions. In the virtual machine, the log file is located at */edx/var/log/tracking.log*. Events are almost any user interaction with the platform, they can be originated either in browser or server side and can be grouped within the following types:

- Navigational Events,

[3]The largest of them all would be the table storing the event logs. Precisely due to its size, it is disabled by default

- Video Interaction Events,
- PDF Interaction Events,
- Problem Interaction Events,
- Open Response Assessment Events, and
- A/B Testing Events.

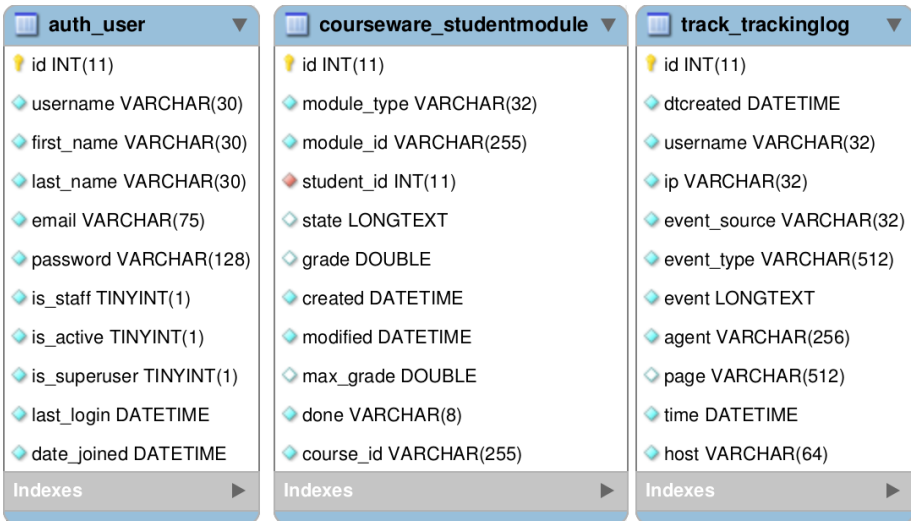

Figure 4.5: User, StudentModule and TrackingLog tables (MySQL Workbench screenshot).

An oddity was noted while familiarizing with the table and inspecting its columns. There are two columns containing date and time information, *dtcreated* and *time*, as could be seen in figure 4.5, where both of them appear defined with the DATETIME type at MySQL logic level. The oddity was that their value did not coincide. The former uses the *auto_now_add* property of Django's DateTimeField. The latter uses the 'event time'parameter of the event logger. They seemed to take the time from different sources with dissimilar time zones. This should be taken into account at the time of utilizing time metrics.

4.2.1 Video information and logs at edX

In the beginning course structure and content at edX were specified using XML. Although most of that data has been migrated to MongoDB, there is still some XML. We are not interested in XML for our analytics, but just a quick review for completeness.

EdX had a tag *video* for the XML to specify these kind of modules. It had an attribute *youtube* to specify comma-separated speed and Youtube video IDs pairs. Typical speeds were 0.75x, 1x, 1.25x, and 1.5x.

After migrated to the MongoDB collection *modulestore*, a document there describing a video would consist of the following keys relevant for videos:

- an *_id* with the serialized course identifier parameters. To identify we are in presence of a video document, we have to inspect the nested key *category* and check it has value video.
- A *metadata* key that provided that the document describes a video, will contain as nested keys: the Youtube identifiers for the videos in all the encoding speeds that it is available, a flag to know if the video file is downloadable from the platform, the video name, source URL (e.g. in Amazon Web Services (AWS)) and a flag to show or hide the captions if available.

In the relational database, tables for models *StudentModule* and *TrackingLogs* (see figure 4.5) keep the information on students interaction with videos. In the case of *StudentModule*, we need to check the *module_type* to look for videos. Possible values of interest are *video* and *sequential*. Sequential elements, which we saw in the course hierarchy of scheme 4.3 and in the screenshot of figure 4.4, are collections of course modules rendered as a horizontal navigational bar in the courseware. They are not exclusive for videos, but can also contain problems and other materials in general. The ultimate video information holders are then those entries with *module_type* set to *video* since the *sequential* option keeps events at container level.

Once an entry in table *StudentModule* with video-related information is identified, that information is mostly found in the *state* column. *State* here is a JSON-like field thus making it flexible for the variety of module types to store their state in the most semantic, convenient way [4].

So far we have seen the video information that is included in the tables active by default. With this data this project could not have been carried out as it relies on event raw data. Our richest raw materials provider is therefore the table for *TrackingLogs* model.

It could be guessed from the event categories we saw in the previous section, that Video Interaction Events, as the name implies, have the video usage or consumption information we are interested in. They are the event that fire when users work with video resources. The first event is server-generated and triggers when the user accesses a page containing a video. It is the *load_video* event and indicates that video is ready to be played.

The *event* field contains information on the particular event in key-value pairs. Play, pause and stop events need no further explanation. Stored event details for these actions are in table 4.3. Until June 25, 2014, a *pause_video* event was emitted when the end of a video was reached at playing. From that date on, it was substituted by the *stop_video* event, created for that purpose.

The *seek_video* event refers to the user clicking the playback bar, moving the

[4] http://edx.readthedocs.org/projects/devdata/en/latest/internal_data_formats/sql_schema.html#state

Key	Data type	Description
id	string	Edx video identifier.
code	string	Video's Youtube identifier.
currentTime	float	Event video position in seconds.
speed	string	Video speed.

Table 4.3: Event field keys for play, pause and stop events.

playing point or clicking the transcript in order to change the position in the video file. Details of subfields in table 4.4.

Key	Data type	Description
old_time	float	Previous position.
new_time	float	Selected position.
type	string	Navigational method used.

Table 4.4: Event field keys for seek event.

The *speed_change_video* event accounts for the use of the playing speed change option. Events keys in table 4.5.

Key	Data type	Description
current_time	float	Video position.
old_speed	string	Previous speed.
new_speed	string	New selected speed.

Table 4.5: Event field keys for change-speed events.

To calculate the percentage of video seen using edX's default settings the single choice is to use the *state* column of *StudentModule*. The problem being that what it is actually registered there is the video position where the student left the player the last time he/she was working with it. The purpose of this state storage is far from analytics, it is that users find the resources in the same state they left it off. Its value does not imply that the student has viewed the video up to that point. Therefore, it is unreliable as a measure but the best that can be done in the default-settings scenario.

4.3 Django basics for this project

"For perfectionists with deadlines", claims the motto of Django, the web framework over which edX is built. Django is the starting point for developing something on or for edX. Familiarity with Django is a must.

At the core of edX there are two Django projects: the Learning Management System (LMS) and the Content Management System (CMS). The former is the learning environment for the released courses and therefore used for students and teachers. The latter is a course authoring tool for content and course creators, typically teachers.

Each of the two projects is comprised of tens of Django applications, many of which are shared or common for both of them. In the case of the LMS there are tens of them. Just to name a few: certificates, courseware, dashboard, mobile_api, open_ended_grading, shoppingcart, verify_student... Their names give an idea of what the tasks they are in charge of are.

File	Description
__init__.py	Marks Python package.
settings.py	Django project configuration.
manage.py	Command-line utility to interact with Django project.
urls.py	"Table of contents".
wsgi.py	For WSGI-compatible webservers.

Table 4.6: Default Django project files.

Parameter	Purpose
DATABASES	Specification of databases to handle with Django.
INSTALLED_APPS	Django applications included in the project.
ROOT_URLCONF	Path to the file containing the project URLs.
TEMPLATE_DIRS	Path to directories to search for templates.
TEMPLATE_CONTEXT_PROCESSORS	Callables to populate template contexts.
STATICFILES_DIRS	Path to directories to search for static content.
COURSE_ID_PATTERN	Regular expression for edX course identifiers.

Table 4.7: Django setting parameters of interest.

Django applications provide modularity to Django projects. A project is a collection of applications with common settings. These applications are pluggable, that is, they can be reused in other projects. At a minimum, Django projects contain the

files listed and described in table 4.6. Notice they all have extension *.py* as Django is written in Python. There are many available settings variables, of which table 4.7 gathers those relevant for this project. All the settings in the table are Django default, except the COURSE_ID_PATTERN that is specific to edX. Static content in edX include, but is not limited to, SASS (Syntactically Awesome Style Sheets), CSS, coffee, Javascript and images.

The basic files internal to applications are in table 4.8. Views are responsible for the user interface.

The metaphor of "table of contents" in table 4.6 parallelizes searching for a content and finding its page with searching for a URL and finding the function in charge of its content.

File	Description
__init__.py	Marks Python package.
views.py	Handles the web application interface.
test.py	Application test code.
admin.py	Objects the admin interface should have.

Table 4.8: Default Django application files.

Database back-end	Python
Table	Model class.
Table entry	Instance of the model class.

Table 4.9: Django's database abstraction.

MySQL table	Django model	File location
courseware_studentmodule	StudentModule	courseware/models.py
track_trackinglog	TrackingLog	track/backends/django.py
student_courseenrollment	CourseEnrollment	student/models.py

Table 4.10: Relevant Django models at Open edX.

Django's database abstraction matches database and Python elements according to the correspondence shown in table 4.9. It is a rather simple table yet it encloses the powerful abstraction philosophy. A table in the database is defined by a Python class that inherits from the *Model* class. A record in the database is an instance of the class defining the model.

Following that abstraction, the MySQL tables at edX most relevant for the project, are gathered in table 4.10 with their equivalent Python class representing

the Django model. Table name is created by the union of the application name and the model class name in lowercase, by means of an underscore. Thus for example, the *track_trackinglog* table corresponds to the *TrackingLog* model defined in the Django application *track*.

4.4 Configuration

In the devstack, edX production settings are overridden by the file devstack.py in edx-platform/lms/envs folder to tailor the project for development and take advantage of Django's debugging features. It is not recommended, however, to customize devstack settings by modifying that file. To do that, it is suggested to create a new settings file that inherits the provided one and assigns in it the new desired values. The server should be run then with that file explicitly declared using the option settings. This is the way this project is run as was shown in figure 4.1, where it can be seen the name of the settings file is *xinsider*. Options changed by the latter come in table 4.11 with their new values. The rest are those inherited from devstack settings.

Parameter	Set value	Default value
INSTALLED_APPS	+= ('xinsider',)	—
TIME_ZONE	'Europe/Madrid'	'America/Chicago'
ENABLE_SQL_TRACKING_LOGS	True	False

Table 4.11: Custom settings for devstack.

MySQL Workbench is a valuable GUI (Graphic User Interface) tool for interacting with edX MySQL database, check the tables and see first hand how tracking log events are stored since some of them are either not fully specified in the documentation or with a different saving format to that indicated there. The parameters' values for the MySQL Server Connection are in the screenshot of figure 4.6.

The SSH access parameters are the same for the analogous GUI tool for MongoDB, Robomongo, whose settings appear in figures 4.7 and 4.8.

There is an issue though with version 2.6 of MongoDB that prevents Robomongo from successfully connecting to the edX non-relational database. That is the MongoDB version included in edX devstack by the time of the latest stages of this project. The configuration parameter *bind_ip* indicates to the Mongo server the IP address from which to listen for connections from applications. It used to be all interfaces by default but in version 2.6 it was set to *127.0.0.1* [61]. It is not possible to change that value in the devstack as permissions are not granted to do that. As

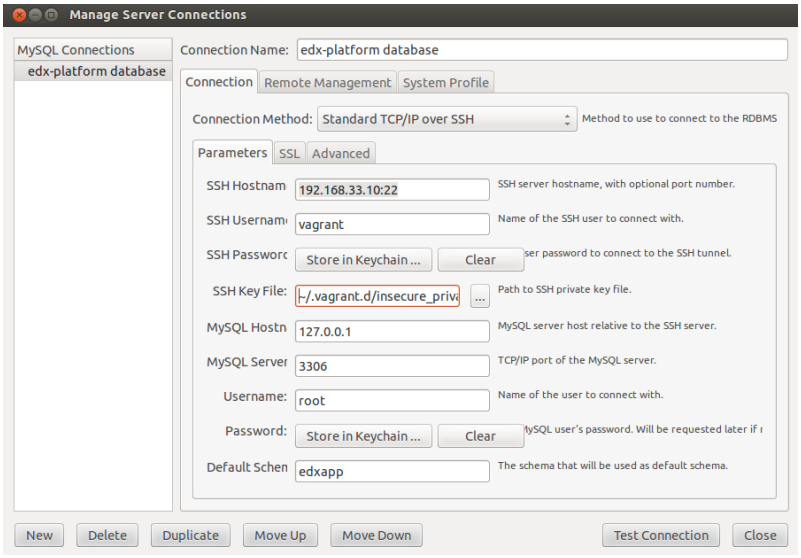

Figure 4.6: Screenshot of connection parameters from MySQL Workbench.

a result, Robomongo was unable to connect to the Mongo server in the platform version using its version 2.6, as prompted in figure 4.9.

4.5 New navigation tab

A new tab is created in edX navigation menu to access the learning analytics visualizations. It has been named *Xinsider*. At edX, the tabs are defined in the file: *devstack/edx-platform/common/lib/xmodule/xmodule/tabs.py*.

Code is added there to specify the new tab. The key *xinsider* with value *XinsiderTab* is appended to the *sub_class_types* Python dictionary in the *from_json* static method from *CourseTab* class.

XinsiderTab is the name of the Python class that will represent the new tab. It will inherit from *EnrolledOrStaffTab*, the "abstract class for tabs that can be accessed by only users with staff access or users enrolled in the course", which in turns inherits from *CourseTab*. The class will also associate a URL to the tab.

Once the new class is defined, it must be instantiated in the *iterate_displayable* method of *CourseTabList* class, together with the condition to ensure access constrains that guarantee the privacy policy.

If a course is accessed in the devstack, the new tab *Xinsider* will display as in figure 4.10, where the mouse is over it, though if it is selected the server will return "HTTP 404 Not Found" error as its URL is unknown to it.

In a Django project, the server will look for URLs in the file pointed to by the setting parameter ROOT_URLCONF, as was presented in table 4.6. By default

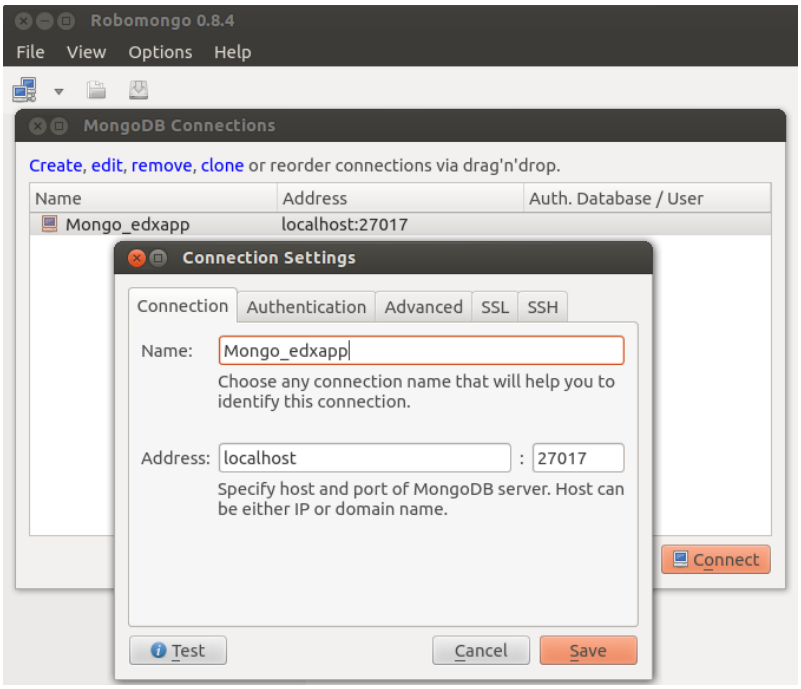

Figure 4.7: Screenshot of connection parameters from Robomongo.

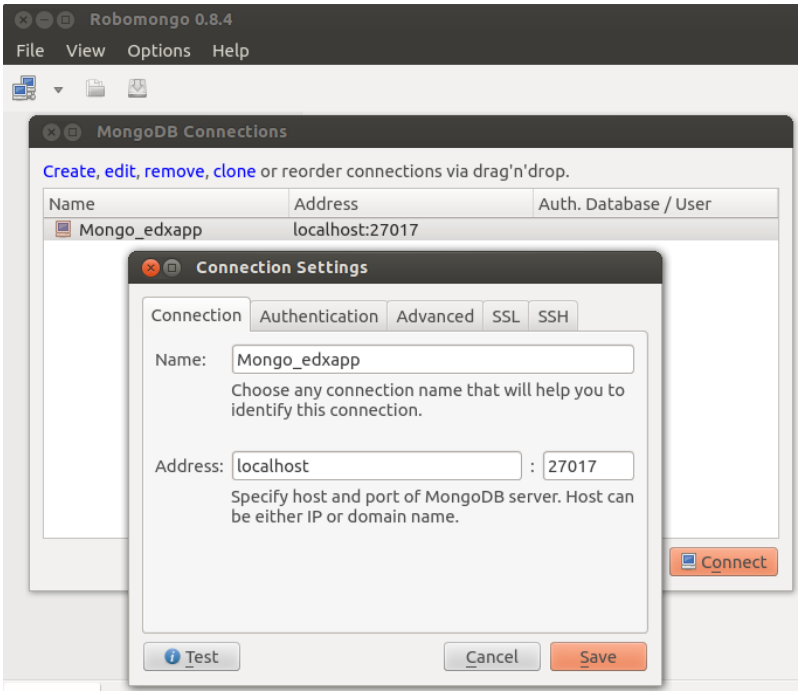

Figure 4.8: Screenshot of SSH access parameters from Robomongo.

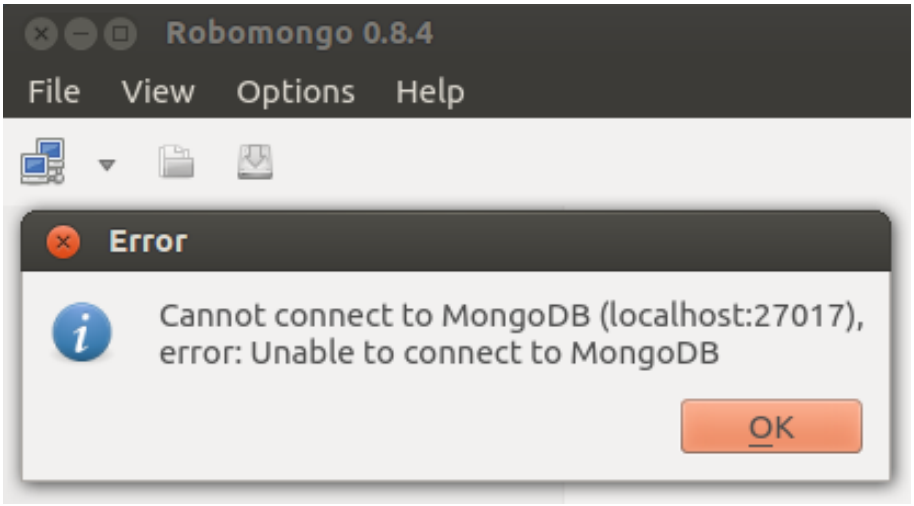

Figure 4.9: Screenshot of Robomongo connection error.

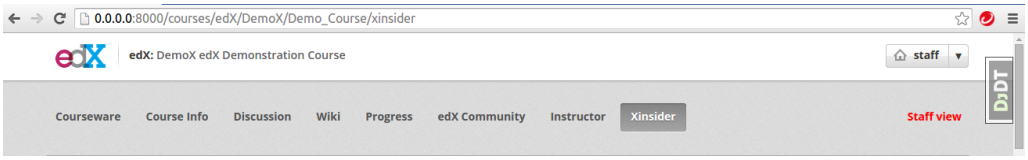

Figure 4.10: EdX navigation menu with *Xinsider* tab (screenshot snippet).

it is urls.py in the project root folder. This is the case of edX's LMS project: ROOT_URLCONF = 'lms.urls'. There, URLs patterns are declared using regular expressions. In edX, URLs usually contain the course identifier. Therefore, they will vary with courses. The pattern for the *Xinsider* tab is as follow:

r'^courses/{}/xinsider$'.format(settings.COURSE_ID_PATTERN) .

The *r* preceding the enclosing single quotation marks delimits a Python regular expression. To assist with regular expression understanding or consultation to the extent relevant for this project, table A.1 has been created.

COURSE_ID_PATTERN is an edX-added setting as we saw in table 4.7. Its value is set to:

'(?P < *course_id* > [^/+] + (/|\\+)[^/+] + (/|\\+)[^/]+)'.

Finally, the server is indicated a function responsible for handling the request, once this pattern is matched. That will be the *index* function in the views of a Django application to be created for edX's LMS. This application is the core of this project contribution to the Open edX Platform.

4.6 Xinsider: the new Django application

In figure 4.8, we saw the files comprising a Django application by default. For us, the most important of them will be that containing the views. It will govern all the

processing and templates framing how results are displayed to the user.

We have called the main function in the views *index*, giving the idea of a starting point when looking for something. Even before the first word of code is placed, we will select some Python decorators to protect access and prevent some accidental or malicious web attack for phishing. Let's recall this application will deal with data about interactions of students with the platform, and this data should not be seen for anyone else that the student itself or the course teacher. A third role with access granted would be researchers. Data packages for research are often distributed to data czars from the participant organizations, but based on the principle of data anonymity.

4.7 Python decorators for Xinsider

In Python, decorators are callable that are passed a function as an argument and in turn, another function, the "decorated" version of the argument, is returned from them [16]. They serve as a resource to comply with the DRY (Do not Repeat Yourself) principle, defining a function for common tasks other functions will need. Writing decorators "can be complicated" [16] but luckily for us we will not need to code them as what we want are common tasks often required. There are three decorators we will apply to the main function of our application.

The first one is the Django provided *login_required* decorator. As the name implies, it aims at guaranteeing that access is restricted to logged users. If the URL of our view is requested without logging in to the platform, this decorator will redirect the user to the page where he/she can sign in. After successful logging, the requested URL will be displayed instead of the dashboard view that normally follows logging, and that was shown in figure 2.3. With the use of the *login_required* decorator we can code without worries for logged access.

A user that has signed up for the platform, however, may left in the browser's cache the content they viewed without being aware of that depending on browser configuration. Even being aware, it may be the case that you do not have the privileges in the system to change the browser's settings. So we must implement functionality to make sure cache policy goes in a deterministic way independent of browser configuration. That comes with the second decorator we will use, whose name is not surprisingly *cache_control* and which also ships with Django. This decorator supports each and every one of the HTTP directives for cache [5]. Options in our interest are: *no_cache*, *no_store* and *must_revalidate*; the three of them set to *True*. That way, in no case cache will be a source of leakage.

Finally, the third decorator is *ensure_valid_course_key*, written by edX. This is for those functions that take as argument a course identifier, to ensure it is correct or raise an HTTP 404 error if it is not. Let us recall that we take the course identifier (*course_id*) from the URL by means of regular expressions, as we saw in section

[5] http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

4.5. This variable will be available then to the function assigned to handle the URL request if it is included in its arguments.

Also available in these terms is the Django *HttpRequest* object, wrapping request data.

4.8 The Xinsider templates

The ultimate task of the function called when a URL is re quested is to render a template that produce the HTTP response with the web page. Therefore, prior to seen what the function must do, we will examine the template containing what we want to achieve or display, and after that we will be in conditions of examining the tasks of the function and the others created to support its purpose.

Though Django has a template language, edX uses the Mako template language. Mako is self-defined as "hyperfast and lightweight templating for the Python platform" [60].

EdX stores the templates for the LMS in the folder edx-platform/lms/templates. There we will create a folder for the templates to be used with our *Xinsider* application. We will start by the template *Xinsider*, whose rendered version will be displayed if the *Xinsider* tab is selected. First of all, we need to guarantee that the general environment does not change as what we are willing to do is an integration. Therefore, the interface must remains the same except for the page content that will show our visualizations. That means the framework of the page has to be the same, including the top band containing the edX logo and course name at left and the access dropdown menu at right with help and log out options, the navigation menu right below, and the bottom menu with links to About, Jobs, News, FAQ and Contact, as well as the copyright, Terms of Service and Honor Code and the Privacy Policy. This is accomplished by inheriting from the LMS *main* template and including the *course_navigation* template marking *Xinsider* as active page.

The main content of the templates will be the visualizations, so we need to check what they require, which library will be used and what are the parameters for them to produce the result we expect.

4.9 Xinsider: the charts' types

We saw in chapter 2 that the progress graph in edX is made using a combination of Python and Javascript. As we need several chart types, we will choose a chart library for the purpose of our project.

There are tens of JavaScript charting libraries available, at least 50 [58]. You are likely to get lost trying to make a decision. You may think at first that you need a table with their feature to pick one, but quickly change your mind when you look at it due to the amount of information. That may be the case if you consult the

table Comparison of JavaScript charting frameworks on Wikipedia [6], several scrolls vertical and horizontal. D3[7] and Highcharts[8] are the two main players according to [14] and [58].

In section 2.3, we saw the precedent of a learning analytics module for Khan Academy. Khan Academy uses Highcharts for their visualizations. However, in that project, Google Charts was chosen in detriment of Highcharts based on licence issues [41, p. 61]. Following the same reasoning and taking into account that Google Charts lacks nothing we need to carry out this project, the latter will be the one.

Google claims its chart library is free, simple to use, cross-browser compatible and cross-platform portable with no plugin required as they are based on HTML5/SVG technology (switching to VML for old IE versions); they also guarantee three years' backward compatibility [9]. To start working with it, it is only necessary to load the AJAX API, i.e. Google JSAPI, then the visualization API, version included, and the chart packages we will be using. Data is not sent to any server, but processed and rendered in the client side.

Visualizations should be given an identifier that will serve to place the chart inside an HTML tag holding that identifier. Chart size may be indicated in two ways, as a chart parameter on the one side and in HTML on the other side. Both ways present advantages and drawbacks. We have preferred the HTML approach to avoid the impact on the navigation experience if the page jumps around while loading due to delays or the order in which it is rendered[10].

Having chosen the library for our visualizations, it is time to examine the chart gallery to match the requirements.

For the *Video time watched* visualization the natural choice would be a column chart as it consists of different videos of which we are willing to know which percentage was watched without segment overlapping and in total. It is then conceived as a column chart with two columns per video.

Video and problem time distribution visualizations offer identical features, being the only difference the data they will show, coming from different resources, namely, videos and problems, and with different metrics therefore to determine time spent on them by students. As each refer to a total, which is the time the student has devoted to that kind of course resource, and requirements tell of interest in knowing the parts of that total, options within the gallery would be: stacked column or bar charts, pie or donut charts (which are the same except for a hole at the center), and the candlestick chart. Though all of them are valid choices, we think that pie or donut chart will help better convey the message.

For the *Repetitions per video intervals* visualization the following are eligible:

- Column Chart: a column for each interval and the repetitions indicated by

[6] http://en.wikipedia.org/wiki/Comparison_of_JavaScript_charting_frameworks
[7] http://d3js.org
[8] http://www.highcharts.com/
[9] https://developers.google.com/chart
[10] https://developers.google.com/chart/interactive/docs/basic_customizing_chart?#sizing

column height. Drawbacks: column width is fixed for all the columns of the chart, meaning that intervals of different duration will have the same column width, although interval is explicitly labelled on the horizontal axis, the visual information the chart conveys is not proportional.

- Histogram: this chart is built automatically from the data it is passed. From the list of values, it determines their frequency. That implies you cannot pass to it the points delimiting the intervals on one hand and the repetition for each fragment on the other hand, but the intervals themselves repeated so many times as they were watched. It looks like a tongue-twister so let us try some example to clarify the problem, or better, reuse the example for the requirements in chapter 3. There, in table 3.1, we have the information as we would like to deliver it to the chart, the intervals and the number of times they were repeated, which is optimum in terms of amount of data to send from server to clients. For the histogram, and that example, we would pass the following data set with the intervals: {0-5, 0-5, 5-10, 10-17, 17-18}. The histogram chart will process this information to determine that the first interval appears two times and the rest a single time. This has the inconvenience that the dataset grows proportionally to the number of repetitions. It would be better to pass the info in the form of two lists, a first for intervals and a second for number of times. This has the benefit of a fixed amount of data. For high number of repetitions the change will be in this number itself, but not in data size to send from server to client. Another drawback is that we do not have the possibility of seen the video length in the horizontal axis, which would be visually more meaningful. Non-contiguous intervals could appear side by side in the histogram if there is in the middle some segment students skipped and video duration will give the sense of being shorter than what it really is if a fragment including the video end was not viewed. The latter is the case of the example, where the video was thirty-second-length, but the interval from second 18 to the end was not watched by any user and therefore does not appear in the information for the histogram. From this analysis we can conclude that although the histogram may seem the most natural choice for a visualization showing repetitions or frequency, it really is not for this particular purpose.
- Bar Chart: this is a 90-degree-to-the-left rotated version of the column chart. Therefore, the analysis is similar.
- Timeline: this chart has a time horizontal axis by definition. Although its purpose is more oriented at showing dates, it could be configured to display only hours. If we set the axis to start at midnight, which is when the day begins, it will start from 00:00, making with this trick that video duration and day hour both coincide. The resolution with this chart would be of a minute. This resolution may result poor for videos lasting only for a few minutes where a one-second resolution would perhaps be desired. To implement the *Repetitions per video intervals* visualization with Timeline, we would use a single row

containing the interval division with colors, and labelling this intervals with the repetition number. This option presents two visual drawbacks. The first one is that for very small intervals, the label may not display correctly as there may be not enough space. This may be solved by placing the numbers above intervals but posing a similar problem for two consecutive very small intervals where their repetition number may overlap. We know that never the solution of a problem should result in another problem bigger or even comparable to the original one. The second visual inconvenience is that the visualization will present the profile of a plain horizontal bar, which do not fit with the irregular profile we are looking for where popular video intervals are easily identified by peaks without having to look necessarily at the numbers.

- Stepped Area Chart: in this chart we have that steps are contiguous, which fits with the idea of a video duration. Step's height can express the number of times intervals were watched. A drawback is that similar to the columns in the Column Chart, steps have all the same width. This disadvantage could be overcome, however, by atomizing the steps, i.e. narrowing them down as much as possible or to the smallest meaningful interval resolution. An interval would be conformed by several of these "atomic" segments, but as their are contiguous and continuous, and every segment in an interval will have the same height, they will simply look as a single step. With this technique we could achieve the variable-width column or step that is desired for this chart as interval are not homogeneous but delimited by the user interaction with video. The "atomic" segment size would be one second as this is the smallest size that would be meaningful for interpreting the visualization. Nobody would be either interested in or capable of distinguishing what happened at a certain millisecond in a video. If we talk about video, the natural resolution for humans will be one second without any doubt. The Stepped Area Chart is our choice for this visualization.

Daily time on video and problems visualization should display time spent on these two type of course resources on a daily basis, as the name itself implies. Days' axis do not need to be either continuous or even contiguous. It is very unlikely that students spent some time on the course daily, including weekends. Possible chart choices are Column Chart, Bar Chart and Timeline; using for the latter tricks similar to those discussed for the previous visualization. Nonetheless, the Column Chart is clearly the natural election for the scope of this visualization.

Finally, for the *Video events distribution within video length* visualization, the original idea was something like a heat map reinforcing with darker colors those intervals where a certain video interaction event appears the most. This approach posed some issues. First and most important, Google Chart Gallery does not include heat map charts. We found a heat map[11] based on Google Charts, that we were examining to a certain extent. We were discouraged to use it as Google does not

[11] http://informatics.systemsbiology.net/visualizations/heatmap/bioheatmap.html

guarantee the compatibility, matching of standards and maintenance for third-party developed charts. In addition, its appearance seemed likely to break the homogeneity of the other charts in the project. Apart from this logistic issue, heat maps are not suitable for color-blind people, leading to a restriction in web accessibility. Among the charts available in the Google Charts Gallery are two eligible for this graph: Bubble and Scatter charts. Bubble charts assist data set visualization with two to four dimensions: two coordinates, color and size[12]. Video events could be shown with this chart in a single, constant ordinate along video duration telling between events by color. Similar close events may be grouped to form a single bigger bubble according to the number of that event's occurrences included. This would raise questions as the vicinity definition to associate events. In addition, as some information would be included in the color, same as above, accessibility issues appear for color-blind people. Bubble overlapping may result in that some appear hidden. On the other side, Scatter charts seem the natural choice to convey the idea of event dispersion or concentration. Different events are better represented in separated ordinates.

For the sake of clarity, chart types chosen for the different visualizations of the project are summarized in table 4.12.

Visualization name	Google Chart type
Video time watched	Column Chart
Video time distribution	Pie Chart
Problem time distribution	Pie Chart
Repetitions per video intervals	Stepped Area Chart
Daily time on video and problems	Column Chart
Video events distribution within video length	Scatter Chart

Table 4.12: Google Chart types selected to implement visualizations required.

For each of these visualizations we will create a JS script that will be included in the main template for our Django application. A Mako template will also be used for those scripts making more modular and easier to maintain and debug the code. Using Mako language, the arguments for the templates will be passed as if they were functions. Basically, the argument they required is the data set.

Templates for the visualizations will have the same code sequence, enumerated below:

1. dataset is passed as argument for the template;
2. a callback is set to run once the Google Visualization API loads, that is, the name of the function charged with drawing the visualization is specified;
3. that callback function is written following the recipe of the remaining steps;

[12]https://google-developers.appspot.com/chart/interactive/docs/gallery/bubblechart

4. the chart type class is instantiated and passed as argument the HTML element that will hold the visualization, typically a ;
5. a control statement is added to check that the data set actually contains data; otherwise, a *"No data to display"* message will be displayed in the space of the chart, replacing the element that has a predefined size suitable for the chart, with a <p> element, thus preventing unused, meaningless, excessive blank space from displaying and impacting user visual and navigational experience;
6. the DataTable object is created from the dataset; this is the format for the chart's API to correctly interpret data; we will below examine this step in more detail;
7. options are declared using the key-value pair format of a dictionary; and finally,
8. invoke the *draw* method with the DataTable created and options to render the chart.

Let us look in more detail to these DataTable objects, indispensable data containers for chart rendering.

4.10 Visualization dataset format

DataTable objects support as column types: boolean, number, string, date, datetime and timeofday; in JavaScript format.

There are several ways in which a DataTable can be populated. We need to understand and compare them to determine the most suitable format for our application to send data to the browser for chart rendering on the client side.

These options are[13]:

1. by means of the DataTable methods: *addColumn()*, *addRows()*, *addRow()* and *setCell()*, whose names are very illustrative of what they actions are;
2. defining the data array explicitly and convert it to DataTable using the *arrayToDataTable()* function;
3. using the JavaScript literal initializer with the table constructor; and
4. querying a data source.

The first method relies on the speed of the browser. The cited functions are not callable from Python and therefore the data set would have to be passed to the client in some format so that it could later go through it with a loop for adding columns and rows, or add the empty rows and then populate them cell by cell instead, which will translate into a larger number of sentences.

The second option creates and populate the DataTable object in a single instruction, with the added benefits of readability and simplicity. The argument is expected

[13]https://developers.google.com/chart/interactive/docs/datatables_dataviews

to be a Javascript array. Therefore we can create the array in our Python-based application, convert it to a format that is correct JavaScript, send it to the client and pass this single argument to the chart. There is a convenient yet simple format to use here for data interchange, the JavaScript Object Notation, more commonly known by its acronym JSON. This is a very clean and efficient procedure that speed the DataTable population process up. No disadvantages were found.

The third method main virtue is processing speed especially for larger tables, over a thousand cells, at the expense of a cumbersome, prone to typos code that is similar but not identical JSON syntax, leading to often confusion to read and write it right as well. Therefore, the trade-off is not very promising as it does not comply with the desired and very recommended simplicity principle.

Finally, the query option would return a DataTable populated with the information requested through an SQL query, but that would require "a web service that supports the Chart Tools Datasource protocol"[14]. In other words, using this method to solve the problem would imply to solve a bigger problem, what makes it really not eligible for our project.

After having examined the pros and cons, if any, of the different ways to provide data to our visualizations, the second option came clear as the choice to make. It is now crystal clear what is the format in which our application should deliver the information for the visualizations.

Let us recall that the visualizations are required to have some selection options, that for students will limit to change the video whose data is displayed in the visualizations where data for a single video is represented; and for teachers will be also possible to select single student data or class aggregation metrics. Nonetheless, there must be data to show by default when the visualizations page is loaded. The criteria will be to show data for the first video in the course, where video-selection option is available; and for a class aggregate where student selection is available, i.e. for teachers only. All visualizations have a single class aggregate, except the video interval repetitions one, with two. The one to show by default out of the latter is that aggregate where single students contribution to the general repetition number is not limited to a single time.

Criteria election reasons are courseware order for videos and collective information for teachers, that we think provide a better whole picture of the indicator, being more meaningful to the learning process and for the purpose of the course outcomes.

Next, we reflect on how to implement these selectors and what they will require from the application.

[14]https://developers.google.com/chart/interactive/docs/queries

4.11 Visualization selectors

There are two interactive features native to Google Charts that could help us implement the selectors we need: dashboards and controls[15].

Dashboards are an automatic way of working with multiple visualizations that shares the same data. They ensure data is shown consistently and coherently, assuming for us the coordination logic among visualizations. In our visualization set, however, there are only two of them built upon the same data. Those would be *Video time watched* and *Video time distribution* ones, where the latter is another way to display the total time viewed per video in a donut chart. This fact of its scarce application to only a third of the visualizations makes our interest in Google Charts' Dashboards to diminish.

Controls on their side are appealing interface filters for the user to select a criterion to visualize a subset from the chart dataset. Available controls are: *CategoryFilter*, *ChartRangeFilter*, *DateRangeFilter*, *NumberRangeFilter*, and *StringFilter*.

CategoryFilter is a dropdown menu to pick an option among a number of predefined values.

ChartRangeFilter consists of two sliders to narrow down or widen the span of a continuous range of values. Similarly, *DateRangeFilter* is a particular case of dual-slider control especially designed to filter a range of dates with both starting and final date. *NumberRangeFilter* can also be guessed to be a particularization for numbers.

Finally, *StringFilter* is a text input box to limit data to that matching the string entered for the defined field.

After this quick overview we can see that those useful for us could be the *CategoryFilter* and *StringFilter* controls. As a dropdown menu, the former is suitable for both student and video selectors as well. The writing text filter has the drawback that the teacher must know the names of students, no much to ask if the case of a traditional class using the MOOC for reinforcement of complement, but very unlikely if not impossible in case of a massive course with thousands, ten-thousands students or over. Dropdown menus are not strangers to this situations as for them a huge amount of students impact the navigation experience as well, regardless that the teacher needs not to know the student names as those are listed, but in a who knows how many scrolls menu.

We should not forget, however, that these native chart controls act as dataset filters. That means the chart needs to be passed all the huge amount of information for all of the students and all of the videos in the worst case, ending in a data overloading that could impact highly the browsing experience and significantly increase visualization rendering latency. Most of these data is likely to turn out to be completely useless as a teacher will not have the time to selecting each and every one of the options, especially when there are a few of them. That makes no sense.

[15] https://developers.google.com/chart/interactive/docs/gallery/controls

The reasonable approach would be to send to the client only data on demand, except for the case of the first predefined screen with criteria stated at the end of the previous section.

Apart from the native Google Charts controls, we can implement the visualization selectors with HTML and JavaScript. Advantage lies in simplicity and layout flexibility as we control where the selector will be placed on the page. This was the technology we finally decided to use.

All of the visualizations have a student selector and two of them a video selector; so selectors will appear multiple times in the code. It would not comply with the DRY (Do not Repeat Yourself) principle to replicate their code. Here, templates come handy again. We will design two templates, one per selector type. Then, in the main template, *Xinsider*, they will be included so many times as needed, after passing them the proper parameters and rendering them.

Student selector template will require two arguments: an identifier and the list of students. As for students this selector should not appear, it will be controlled in the template if the user for whom the template is rendered has staff access. In addition, a second control will check for a blank student list. If the latter is the case, a "No students enrolled yet" message will display instead of the selector. Provided that student list is not empty, the first menu option will be the class aggregated value, which we will call "Student average". Student list will be grouped under the label "Students". An option will be added for every student by looping over the least.

Video selector template will be passed three arguments: an identifier for the menu, the video name and the video identifier. Video names have none unique constraint imposed. A handful of videos may share the same name. The name of a video when creating the course may be even left blank. That is why the identifier is required, as it plays an indispensable role. Names will be displayed on the menu while the *value* attribute will store the correspondent video identifier. The identifier is not suitable for the menu as it consists in part of an alphanumeric code. Needless to say, that code is not human readable. Therefore, it makes no sense to display video identifiers on the menu entries. In case there are no videos in the course, the message "This course contains no videos" will display instead of the dropdown selector.

There is an oddity not addressed by the templates. Video interval repetitions visualization requires two aggregates included in the selector. We will use the JavaScript DOM model to make some adjustments. In the menu for that visualization, the "Student average" entry will be removed, and two options named according to the aggregates they represent will be inserted at the top of the selector: "Class: total times" and "Class: 1 student 1 time". Finally, a semantic change is also undertaken. For the video events visualization, the "Student average" entry name makes no sense for the aggregate. There, it would be more meaningful a name like "All students".

At this point, the page layout is finished. The use of templates have made the code cleaner, modular and reusable. It is still needed to add functionality for a

change in the menu to result in a change in the respective visualization.

The idea of dynamic visualizations whose data is changed on demand points to an AJAX implementation. "AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page" [56].

4.12 Data on demand via AJAX

AJAX stands for Asynchronous JavaScript and XML. It is a technique that helps design fast and dynamic web pages using existing standards [56]. To implement Ajax there is a JavaScript library to make the work easier: jQuery.

> jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.[16]

We will need neither to download jQuery nor add it to our web application after it as it is already widely used at edX and included in the templates inherited by ours.

For the principle of modularity, we will implement the Ajax functionality using jQuery in an external file included by our interface template.

We start by assigning events handlers to the change event of our dropdown selectors. Actually, all of them will have the same handler, a JavaScript function we will define with name *updateChart*. For the student selectors a control will be placed to assign the handler only for staff users. This is not at all necessary since the selector themselves will not appear to non staff users, due to the control sentences we added at that selectors' template. However, a redundancy to prevent unwanted access never hurts and in this case it is a simple instruction.

Handlers are assigned inside a function, which in turn will be registered in the page at the time it loads, that is, when the ready event is emitted.

Now, the *updateChart* function is written. It will receive as argument an *event* object, containing information of the event that invoked the function. Its *id* property will help us identify the selector that the user changed. Selector identifiers were assigned in the template by adding a number to a root name depending on their type: *student_select* and *video_select*. Thus for example, the first visualization, *Video time watched*, will have a selector identified as *student_select1*. Being the number the distinctive element. For the video selectors, as they are only present in two visualizations, their number will be the same as that of the student selector to reduce the risk of confusion when writing the code, that is, for simplicity.

The visualization can then be identified for the number in the selector *id*. To tell between selector types, the strings "student" and "video" will be searched for

[16] http://jquery.com/

within the *id* also. Once uniquely determined the selector, its value is accessed to have an identification of the information that will be requested to the server. We are in conditions of writing Ajax request with jQuery. Table 4.13 is a summary of the fields that are more likely to be included in our AJAX jQuery directive. We will

Key	Description
url	URL for the request.
data	Data to send.
processData	Whether to convert data to a query string or not.
type	Request method: POST or GET.
dataType	Format of data returned.
success	What to do if the request succeeds.
error	What to do if the request fails.
complete	What to do in any case.

Table 4.13: Relevant options for AJAX jQuery directive.

define a new URL for the AJAX petitions. We choose the URL to be the same as that of our application, with the */chart_update* suffix appended. To do that, we will declare the URL in the file where the server looks for them, and that we have seem before it is *lms/urls.py*, the server "table of contents" (see table 4.6). There we will assign the response to be handled by the view *chart_ajax* created for this project and that we will examine in the next section.

The data we will need to include in the request will be the identifiers for: user, video and visualization number. Why not the course identifier? The course identifier is indeed essential to determine the exact dataset that will be sent to display in the visualizations. Course content in edX is reusable, so that same video may be in another course, for which that same user is also enrolled. The fact is that we do use the course identifier, but we do not need to send it as data, as we now it is part of the URL, and that using regular expressions we can obtain it and pass it to the request handler.

The data to send will be converted by default to a query string. This is simpler, though not indispensable, and it is controlled by the *processData* parameter, which defaults to *true*. To convert to string safely, data should be strings themselves or of some of the built-in types. Otherwise, in case we need to send a user-defined data type, *processData* will be set to *false* and data retrieval on server side would not be that simple. Luckily for us, our three data parameters are string ones.

Request *type* is not limited to the options in table 4.13 but GET and POST are the most common methods. GET method is indicated for information request to the server. POST method will be used if as a result of the request, data in the server side will change[17] inserted, updated, deleted or whatever. As our application will

[17]http://learn.jquery.com/ajax/key-concepts/#get-vs-post

only query the server for data to be shown in the visualizations, GET is the method recommended for our purpose.

We have previously discussed the format most suitable for the visualizations and decided it would be JSON. That is also the format more frequently used to receive data from an AJAX query. Therefore, we choose the *dataType* to be JSON.

It only remains the definition of the callbacks to run for the different request results: success, failure, or at request completion regardless of result. Success-handling function is passed the response. This function will call another function to redraw the involved visualization with the received dataset. In addition, for the purpose of debugging, we will log the received package to the console, to be seen on the browser developer tools. This way, we can make sure we received what we expected and in developing time helps correct syntax or semantic errors. That will be the same use of the *error* and *complete* callbacks, to log the error thrown if it proceeds, and the request status in any case.

The redraw function task is rather simple. It uses a switch-case structure to determine from the visualization number, the visualization that needs to be updated and re-invoke its drawing function passing it the new dataset.

And we are done with the client side part of our application. Next, the view created to handle the URL defined to listen to Ajax requests is briefly examined.

The *chart_ajax* view will use the same Python decorators as the main application view, *index*. Let us recall they were: *login_required*, *cache_control* and *ensure_valid_course_key*.

The course identifier in converted from string to *CourseLocator* object. Its is then passed together with the data extracted from the request to a function charged with obtaining the dataset in JSON format. The JSON is the used with the *HttpResponse* Django object, explicitly indicating the type *application/json* for the HTTP MIME type.

4.13 Xinsider template context

The context here is the set of variables and their values that are passed to the template from which Django will build the HTTP Response. It is a Python dictionary. This section deals with the keys that need to be included in the context.

To serve as a guide, they are listed below:

- user,
- course descriptor,
- staff access,
- Studio's URL,
- masquerade status,
- reverification data,
- list of students enrolled in the course,

- video names,
- video identifiers, and
- data sets for visualizations.

The user is determined directly by a property of the request. The template will use it to customize the user interface.

The course descriptor is obtained in two steps from the course identifier in *string* format, available from the URL. First, the course key, a *CourseLocator* object is obtained from the *string*. Then, the corresponding course descriptor is looked up by means of the *get_course_with_access* method of the courseware application. The method is passed the course key, the user to check if they are enrolled in the course, the *action* option set to *load* to load the courseware and be able to inspect inside the course and *depth* option set to *None* to load all the course children. The course descriptor is used in the template for the page title.

Staff access will be true-or-false valued depending on whether the user is staff for the course. This is checked with the method *has_access* of *courseware*.

The URL for Studio, i.e. edX's CMS, allow for direct access from LMS to the CMS for course authors. *Courseware* provides the method *get_studio_url* to obtain it in a single step.

Masquerade allow course staff to see a student or staff view of courseware. JavaScript and Ajax are used to toggle between roles. The state where it was left in the last session is stored. This masquerade variable will contain a string version of the status of masquerading (either 'staff'or 'student'). The masquerade state is queried through *setup_masquerade* function of *courseware* application.

For the feature of certificate emission, edX ask students opting to these achievement certificates to re-verify their identity, to address the issue of to whom actually certificates are granted. Reverifications are implemented in the *dashboard* application template *_dashboard_prompt_midcourse_reverify.html*. That template is included in ours. We need to pass to the template the information to display on the correspondent banner provided that user identity is to be re-verified. This data is retrieved also with a *courseware* method, *fetch_reverify_banner_info*.

The students enrolled in the course can be retrieved using the Django database abstraction. The table containing that information is modelled by the class *CourseEnrollment*. That class has a class method to precisely return a query set of *User*. The class method is *users_enrolled_in* and needs the course key to be passed in.

We can obtain both video names and identifiers from video descriptors. There are at least two ways to obtain the list of video descriptors for a course: 1) using the edX's abstraction for its MongoDB database, the *modulestore*; and 2) iterating over a course descriptor.

The first option would consist of a single line, using the *get_items* function of *modulestore*, to which the course key and the content type, video in this case, are passed. The drawback is that documents in the MongoDB database are stored in no particular order. Therefore, video descriptors and any content in general returned

by *get_items* will have a random order. We need video names in the order they appear in the course, what automatically discards this procedure.

This is not the case of iterating over the course descriptor by definition. Course structure and content follow the hierarchy of figure 4.3. We need to traverse the tree structure from the root (the course) to the leaves (content) by means of the *get_children* method. Thus, from the course we get the chapters; chapters are the parents of sequential (or videosequence) elements, which in turn have the vertical elements as children, that are the last elements in the container chain. From vertical elements hang content. As video is only one of the several types, content category must be checked. It results in a number of code lines, although it is not difficult, but just the nesting of similar instructions until the leaves are reached. In exchange, we have the videos (or any other content) in course order.

Children here are always descriptors. However, from the video descriptors the video identifiers and names are easily obtained. The name is directly the attribute *display_name_with_default* of the descriptor.

Last but not least, the data sets in JSON format to be used directly by the visualizations. In fact, these are the most important variables in the context as their content is the core of this project, the result of the data processing to be visualized. There should be six data sets, one per visualization.

4.14 Xinsider tables

Our application will create new MySQL tables to store the result of data processing for the visualizations. Due to their similarities, the first three visualizations can share the same table, although they will not use all of the fields. For the remaining visualizations a new table will be created. That makes four tables.

Let us briefly review Django database abstraction. Tables are modelled using Python class that inherit from the Model class. Class attributes are the table columns and class instances are table rows. At edX, this holds for MySQL but not for MongoDB; the reason being that Django has native backend support for the former but not for the latter. That is no problem for our project as our tables will meet the relational database criteria, which makes MySQL a great choice.

Model class	MySQL table name
ConsumptionModule	xinsider_consumptionmodule
VideoIntervals	xinsider_videointervals
DailyConsumption	xinsider_dailyconsumption
VideoEvents	xinsider_videoevents

Table 4.14: Equivalence between Xinsider class models and MySQL tables.

The relation between the model class name and the table name in MySQL is

pretty straightforward. The latter will be the name of the application joined with the class name in lowercase, with an underscore in between. Our application name is xinsider as we discussed in section 4.6. Equivalence between the abstraction and the real table names are then as indicated by table 4.14 for clarity.

ConsumptionModule will be designed to store the data required for visualizations one to three. The first column will hold the student identification. EdX uses the Django's User model for user authentication information. It seems natural then to use the entries in that table as foreign keys as in fact does edX for some models. This would help with database consistency as it is not possible to have a student identification that is not registered in edX. We are interested, however, in storing the aggregate values in the table as if they were simply another student's. The underlying philosophy is that no processing or the least is undertaken at the moment of data request for visualization, but that is already done and the only step needed is to query the database.

To avoid collision with existent of possible user names we have to use a name that is not a valid registration name. We tried to find out the restrictions imposed on names by attempting to register with several non alphanumeric characters. A screenshot of the result is in figure 4.11, according to which usernames should only consist of a combination of letters and numbers without any space in between. We tried to register with name "_average", a candidate to use for the aggregate of all students. There was no error prompt in the first registration step as before, leading us to assume that although it is not specified in the user name constraints, underscores are accepted. We do not know for sure, but it is very likely that edX username policy is based on Django. In the documentation for the *username* field of the *User* model, it is stated that they "may contain alphanumeric, _, , +, . and - characters"[18]. Second choice was then to try "#average" that was immediately rejected after clicking the "Create My edX Account". The number sign can then be safely used in the name given for the class metrics.

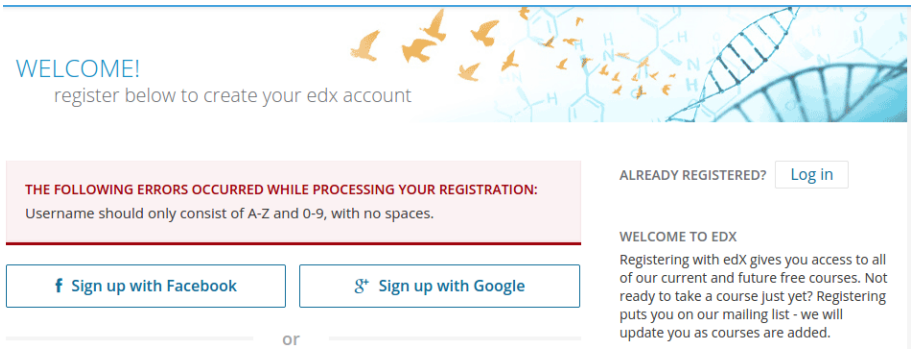

Figure 4.11: Screenshot of user name restrictions prompt.

We have seen the inappropriateness of using *User* as foreign key. A text type

[18]https://docs.djangoproject.com/en/1.7/ref/contrib/auth

will be used instead, Django's *CharField*. The length or number of characters will the set to a maximum of 32 (Django admits 30 at most).

Second field of *ConsumptionModule* is chosen to be the course identifier or key. It could also be stored as *CharField* but edX recommend for this parameter the *CourseKeyField* they have created for this purpose in the *xmodule_django* application. An error will be raised if the course key format is incorrect.

The table will be used for videos (first two visualizations) and problems (third visualization). As a result, a distinction must be made between these resources. The module type is specified with a *CharField* restricted to two values: "video" and "problem". A tuple is created for the restricted choices, making it easier for future changes to the model when other resources for similar analytics may be incorporated. The default is set to "video" in a democratic choice as they win two visualizations to one.

Apart from the module type, it is necessary the module key to uniquely identify the resource. A string or *CharField* would fit perfectly, but edX has developed the type *LocationKeyField* to store this information, similarly to what was done for the course identifier and in the same application. Invalid module identifiers will provoke an error when trying to save to the database.

It follows the problem or video name, naturally a *CharField*, which we have restricted to 255 characters.

The last two columns for the table are both Django's *FloatField* types containing the processing values for the total time on that resource and the non-overlapped time. The latter is only applicable to videos and therefore, its options *null* and *blank* are set to true. They seem similar options but "null is purely database-related, whereas blank is validation-related"[19]. The two of them are expressed in seconds.

For a single student there will be a single row for each particular module, as the metric includes all the time he/she has spent in that module so far. This uniqueness is enforced at database level with the Django option *unique_together*, that here will include the student and the module key. Thus, a *ValidationError* will be raised if an entry with these fields is attempted to save twice to the database.

Adding *__unicode__()* methods to the models is a must in Django for two reasons: "sanity when dealing with the interactive prompt" and "also because objects' representations are used throughout Django's automatically-generated admin"[20]. We have seen that in practice, it is not only important that the models have an unicode method but that this is written correctly or no record will be saved to the database although the latter is created.

The second model is *VideoIntervals*. It has a number of similarities with the *ConsumptionModule* model: student, course key, module key and module name fields, same fields unique together and similar unicode representation.

[19]https://docs.djangoproject.com/en/1.7/ref/models/fields/#blank
[20]https://docs.djangoproject.com/en/1.4/intro/tutorial01

The difference is that at the core of this table are the interval delimitation points and the interval height (or number of times it has been seen). We will name them *hist_xaxis* and *hist_yaxis* respectively, where hist is short for histogram as it is how this visualization intends to look like. Both of them are lists of integers, and thus will be stored as text. Their size is hard to estimate as it could be very big, depending on video length and on intensive user consumption for that video. The *CharField* type requires the maximum number of characters to be explicitly declared. We have rather used *TextField* that does not have this constraint and symbolizes a large text field.

The two aggregates required for this visualization will be named *#class_total_times* and *#one_stu_one_time* in the student column, following the convention of the preceeding number symbol defined for avoiding collision between aggregate names and possible student names.

The third model is *DailyConsumption*, for the fifth visualization. Similar to the former two models, it has fields for the student and the course key. It has a module type as it was the case for *ConsumptionModule*. The module key here makes no sense as time spent on individual modules here is aggregated on a daily metric for module types. The metric fields are a list of dates and a list of time values in seconds for a given module type of the two covered: video and problem. Django's *DateField* field uses a Python's *datetime* object to store data. Databases, however, do not store *datetime* objects, so the field value must be converted into an ISO-compliant date string for insertion into the database. Trying to save a list of *datetime* objects resulted in error, so the decision was to save dates as string using a format as discussed above. The names of the columns are *dates* and *time_per_date*.

The unique together restriction will be imposed on the tern formed by the student, the course key and the module type.

VideoEvents is the fourth and last model. It has also columns for the student, the course and module keys and module names. Student aggregate would not make sense at it would simple be the repetition of existing entries put together. That means here it is possible to use the *User* model as foreign key for the student. It has not be done like that just to keep homogeneity between the visualizations, which facilitates development and code readability.

A column is defined for every video event to store the list of the positions where the event took place. That columns are: *play_events*, *pause_events*, *change_speed_events*, *seek_from_events* and *seek_to_events*. Discussion for list format is similar to the lists in the previous tables. In consonance, their type will be also *TextField*.

The columns to hold uniqueness together are those for the student and the module key.

In the MySQL Workbench screenshot of figure 4.12 it could be seen how model classes and attributes translate to the MySQL table definition. Types are translated into those that can actually be written to the database. Django's *CharField*, *FloatField* and *TextField* are converted to *VARCHAR*, *DOUBLE*, and *LONGTEXT* MySQL types respectively. EdX-defined *CourseKeyField* and *LocationKeyField* are

stored as VARCHAR also in the MySQL tables.

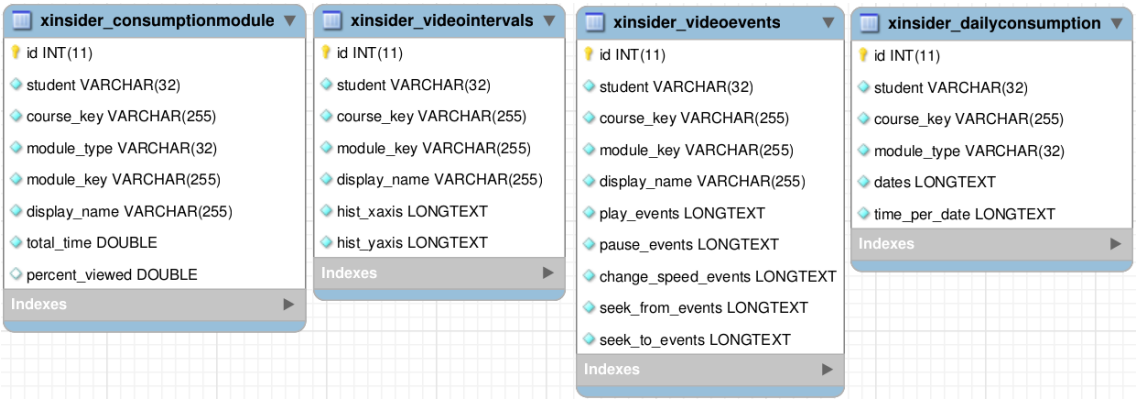

Figure 4.12: Xinsider tables (MySQL Workbench screenshot).

Of course, this four-table, modular approach is not the only option. Given the similarity or identity of two to three columns in all the tables, the design could have been also thought as a single table that allows for *null* and *blank* options whenever a column value does not proceed.

4.15 Data processing functions

It comes the time to deal with data. Previously, we have seen some details in advance, particularly how course and video descriptors were obtained. In the following subsections we will examine the different functions created to process data or obtain information about resources at the time of executing the processing task using Celery as scheduler. Processed data is then saved to database almost ready for a visualization request. Almost ready because although we wanted to separate data processing from the visualization time as it would result in an undesired latency for the user with the correspondent impact on their navigation experience and satisfaction with the module in consequence, a trade-off must be achieved between this latency and a meaningful data storage in the database. What we mean will be more clear in section Data querying (4.16).

As a convention, strings are encoded using UTF-8. Python variables will be encoded to the JSON format we discussed as appropriate for the visualizations using the JSON encoder and decoder *simplejson*. "It is the externally maintained version of the *json* library contained in Python" over which it has "performance advantages" [21].

[21] http://simplejson.readthedocs.org/en/latest/

update_visualization_data

Argument: course identifier.

Returns: nothing.

This will be the function periodically scheduled to process data and update analytics records in the tables. The scheduling is done with the Celery application edX uses. Apart from small settings, this is simply achieved by means of the *task* Python decorator to handle tasks, as the name implies. It is a big and core function that comprises and articulates all the data processing and persistence for the *xinsider* application.

As we saw in section 4.14 there is some information common to several of our tables. To reuse this data and avoid duplication two simple classes with no method other than the initializer are defined: *UserVideoIntervals* and *UserTimeOnProblems*; their attributes are listed in table 4.15. All of them are lists except student name and video and problem identifiers.

UserVideoIntervals	*UserTimeOnProblems*
student	student
video_id	problem_id
interval_start	problem_time
interval_end	days
vid_start_time	daily_time
vid_end_time	
disjointed_start	
disjointed_end	

Table 4.15: List of attributes for classes *UserVideoIntervals* and *UserTimeOnProblems*.

videos_problems_in

Argument: course descriptor.

Returns: two lists of video and problem descriptors for the course respectively.

Traverses top-down the course hierarchy or tree to get these modules in course order.

get_info_videos

Argument: list of video descriptors.

Returns: 1) list of video names; 2) list of video module identifiers; and 3) list of video duration for videos in the course.

Video names come directly from the *display_name_with_default* attribute of descriptors. The same for video identifiers from the property *location* instead. Youtube

identifiers for the videos are retrieved subsequently from *_field_data_cache* and *youtube_id_1_0* nested keys of the descriptor dictionary. Youtube identifiers are the means of obtaining video durations as this parameter is not stored in edX databases. These identifiers are passed to *id_to_length* method to obtain the video duration.

An example of some of this information retrieved for the EdX Demo Course built-in to an instance of the development stack is stated in table 4.16. Video names appear in course order. The storage order of video documents in the MongoDB collection is irrelevant, included just to show they follow no logic order.

Video name	Youtube Id	Duration (s)	MongoDB order
Welcome!	b7xgknqkQk8	195	1
Building a Computer Memory Element	an9jasFZK1c	131	143
Biology Demonstration	gnqth8uWzO8	130	31
Connecting a Circuit and a Circuit Diagram	o2pLltkrhGM	275	156

Table 4.16: Examples of video information retrieved.

id_to_length

Argument: the identifier for a video in Youtube.

Returns: video duration in seconds.

This function uses the Youtube API to obtain video duration. Before any operation can be performed with the YouTube Data API, some setup is needed[22]. First of all, the Google Data Library must be installed. To do that the package has to be downloaded to a folder in edX's developer stack virtual machine. From there, the command: python ./setup.py install will have it installed. Libraries *gdata.youtube* and *gdata.youtube.service* can then be included. An object *gdata.youtube.service.YouTubeService* is to be initialized and HTTPS/SSL access turned on. The method *GetYouTubeVideoEntry* from the object will bring information on that video. Nested attributes *media*, *duration* and *seconds* will subsequently lead us to the duration in seconds, as could be guessed. An important thing to notice is that those seconds come as text and therefore have to be converted to an integer prior to perform any mathematical operation.

During development tests, it was noted that even though a video was wholly watched, a hundred percent in the corresponding indicator was never achieved. We saw empirically that the maximum video position at edX was around a second smaller than the video duration retrieved through Youtube API. The shorter the

[22]https://developers.google.com/youtube/1.0/developers_guide_python

video the more this second impacts the indicator. To compensate this difference, in this function a second is subtracted from video duration and then when calculating percentage in other functions, a control is included to prevent non-overlapped video percentage from slightly surpassing the hundred percent as a result of this artificial adjustment.

find_video_intervals

Arguments: 1) a student name and 2) a video identifier.

Returns: four lists, two for the start and end intervals that a student has viewed in a video, and other two with the timestamps of these intervals.

The procedure in this function is event-driven. Some limitations that impact the interval determination quality for interpretation have to be taken into account, especially two of them:

1. Events scope is limited to the platform. They do not register whether the user changed the active tab within the browser, or the active window, that is, moments in which the video do not appear on screen even though it continuous to play.
2. The fact that the user has clicked the play button does not imply that he/she viewed the video part that was played.

Having this inherent limitation clear, let us start by examining exhaustively the different user-video interaction possibilities and the events involved in them. This is empirically done by performing the sequence of actions stated and looking for the generated events with the help of MySQL Workbench. Table 4.17 contains the cause-effect relationship corresponding to actions and the events they trigger.

After a *seek_video* event (user-driven change in video position) there is automatically a *play_video* event if it was preceded by either a *load_video* or *play_video* event. A *seek_video* event after *pause_video* one does not trigger a *play_video* event.

Table 4.17 indicates that the events we need to fetch to determine video intervals watched are a mixture between Navigational and Video Interaction events, enumerated below:

1. *play_video*,
2. *pause_video*,
3. *stop_video*,
4. *seq_goto*,
5. *seq_prev*,
6. *seq_next*,
7. *page_close*, and
8. *seek_video*.

We can group them into three categories according to video playing:

Interaction	Events involved
User watches whole video without interruption	play_video, stop_video
User watches whole video with pauses	play_video, pause_video, stop_video
User watches a video segment	play_video, pause_video
User changes video position when watching the video	play_video, seek_video
User navigates through the sequential	play_video, seq_goto OR seq_prev OR seq_next
User plays the video and log out	play_video, page_close
User plays the video and changes to another sequential	play_video, page_close
User plays the video and changes browser tab or active window	play_video
User plays the video and closes the browser	play_video, page_close

Table 4.17: User-video interaction possibilities.

1. Events starting a playing: *play_video*.
2. Mixed events, i.e events that both halt and start a playing subsequently: *seek_video*.
3. Events halting a playing in progress: the remaining six.

Video intervals measure involves a starting and a halting video events. The steps required seem to be:

1. Query the database for the eight involved events enumerated above.
2. Clean the list. As playing starts with a *play_video* events, it should be at the top of the list. All the events that appear before the first *play_video* are removed from the query set. If there are two consecutive *play_video* events the first is discarded and the second taken as the good one. This oddity was observed when the video page was loaded and the playing bar was in a position other than video starting. A first *play_video* triggers with *current_time* set to zero and immediately a second *play_video* triggers with the correct position as was retrieved from the state column in the *StudentModule* table.
3. Create two lists for video intervals, for starting and ending position respectively.

Video percentage viewed does not depend on the speed with which the user watches the video. The speed affects the time the user spends watching the video, but in no case the relative position of the marker at the playing bar.

It was observed that the *current_time* attribute of the *play_video* event occurring after *seek_video* is not reliable. Sometimes it takes the value of the *new_time* key of*seek_video*, that would be the correct value, and sometimes it takes that of the *old_time* that would be incorrect.

In the process of examining event behaviour using MySQL Workbench to inspect the *TrackingLog* table we saw that *pause_video*, *stop_video* and the navigational events that halted video playing in progress, triggered a *â/save_user_state* event containing the video position for the *state* column of *StudentModule*. The three dots mean the name of the event is way longer at the beginning (see figure 4.13 for an example of the complete name). This finding led to a great simplification on the list of events needed from the eight we enumerated to just three: 1) *play_video*, 2) *seek_video* and 3) *â/save_user_state*.

With these three events a state machine needs to be implemented in the function to properly address the different event combinations when traversing the event query set for information to determine the video intervals. It is described by table 4.18. The down arrow indicates that events in rows are those at time *t*, i.e. current; and the right arrow signals that column events are those at time *(t-1)*, i.e. the previous event. Column *First* covers the case of the event heading the query set. Start and end refer to each of the interval lists respectively. To control what was the previous

Event ↓ (t) → (t-1)	First	Play	Seek	Save
play	Push *currentTime* to start.	Pop start. Push currentTime to start.	Do nothing	Push currentTime to start.
seek	Never first.	Push *old_time* to end. Push *new_time* to start	Pop start. Push *new_time* to start	Push *new_time* to start.
save	Never first.	Push video position to end.	Pop start.	Do nothing.

Table 4.18: Algorithm for video interval determination from events.

event, a flag for event is created.

Interval start position must be smaller than the end position by definition. There is however a particular case where this is not the case for the algorithm in table 4.18. It is when a video playing has reached the end and either a seek or play video event occur. If the video is played after reaching the end that is taken as the starting

interval position and the end will be smaller than or equal to that value as that is the video length.

Also empirically was detected that if the user changes video position quickly two times, in the middle of the correspondent *seek_video* events a *play_video* event is recorded. This leads to a null interval as starting and ending positions coincide. Therefore this irrelevant interval should be removed from the list for it to be meaningful and as clean as possible.

In practice, the list of the eight enumerated events appears in the *event_type* column just like they were stated. On the contrary, *â/save_user_state* is just the last part of the much more elaborated string defining the event (see figure 4.13). Bad news is that this is also the only field where we can inspect the course identifier to check that the entry corresponds to the course for which we are currently processing the information. The course identifier is here built using a de-serialization of its separated keys that pieces them together with a combination of a semicolon and an underscore, as can be also appreciated in the figure. We said bad news because this will make the database query a more expensive operation.

To assist with building the string to query the database for the course we want and the *â/save_user_state*, the method *to_iterable_module_id* is created.

For the enumerated eight events the information to identify the course is found in the *event* field instead of the *event_type* one. It can be noticed in figure 4.13 that this identifier uses a minus sign to put together the keys identifying the course. We will not need an auxiliary function to build it similarly for querying, as this is already defined in edX and it is the *html_id* method of the module identifiers.

The query requires four conditions to obtain the entries relevant for our video interval calculation. Three of them will be mapped to a *LIKE* SQL statement, which is an expensive operation.

Before processing the query set with the algorithm of table 4.18 two steps must be taken: 1) check the query is not empty and 2) check there are at least two events in the query (start and end) once all the events preceding the first play event are removed.

event_type	event
/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_video;_5c90cffecd9b48b188cbfea176bf7fe9/handler/xmodule_handler/save_user_state	{"POST": {"youtube_is_available": ["false"]}, "GE
hide_transcript	{"id":"i4x-edX-DemoX-video-5c90cffecd9b48b188
/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_video;_5c90cffecd9b48b188cbfea176bf7fe9/handler/transcript/translation/en	{"POST": {}, "GET": {"videoId": ["qWxm7CA2v2
load_video	{"id":"i4x-edX-DemoX-video-5c90cffecd9b48b188
/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_sequential;_19a30717eff543078a5d94ae9d6c18a5/handler/xmodule_handler/goto_position	{"POST": {"position": ["5"]}, "GET": {}}
seq_goto	{"old":3,"new":5,"id":"i4x://edX/DemoX/sequentia
/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_video;_5c90cffecd9b48b188cbfea176bf7fe9/handler/xmodule_handler/save_user_state	{"POST": {"saved_video_position": ["00:05:35"]}
hide_transcript	{"id":"i4x-edX-DemoX-video-636541acbae448d98
/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_video;_636541acbae448d98ab484b028c9a7f6/handler/transcript/translation/en	{"POST": {}, "GET": {"videoId": ["o2pLltkrhGM"]
load_video	{"id":"i4x-edX-DemoX-video-636541acbae448d98
play_video	{"id":"i4x-edX-DemoX-video-636541acbae448d98
seek_video	{"id":"i4x-edX-DemoX-video-636541acbae448d98

Figure 4.13: Sample of tracking logs columns event_type and event (MySQL Workbench screenshot).

In figure 4.13 it is also relevant that the *saved_video_position* key has a value with format HH:MM:SS (hour:minute:second). To convert it to seconds the method

hhmmss_to_secs has been created.

Once intervals are obtained they are sorted by the start list keeping the relative position of the other three lists with respect to it.

This is the equivalent for videos of the *time_on_problem* method.

to_iterable_module_id

Argument: a BlockUsageLocator object, i.e. a module identifier.

Returns: a list of serialized components of the identifier.

Very simple function to build a list by manually including one by one the different attributes of module identifiers.

hhmmss_to_secs

Argument: a string representing a time position with format HH:MM:SS.

Returns: the number of seconds equivalent to that position.

Using a regular expression the argument pattern is checked for correctness. Then the different time units are split and converted to numbers from which the seconds are obtained using the relationship between these units.

video_len_watched

Argument: two lists for interval starting and ending positions respectively.

Returns: the two passed lists with interval-overlapping removed.

The purpose is to obtain disjointed intervals.

time_on_problem

Arguments: two identifiers for 1) a student and 2) a problem respectively.

Returns: 1) time spent on the problem, 2) days worked in it and 3) time for each of the previous days.

Time devoted to the problem will be computed as the time problem is displayed in the course, that is, from the moment the problem is loaded up to the instant where user navigates to another courseware resource. Problem loading is indicated by the *.../problem_get* event. The three dots indicate the event type recorded is much longer, as can be seen in figure 4.14, where a sample of Navigational and Problem Interaction events is captured in MySQL Workbench.

The events that could put an end to the problem time measurement are all of type Navigational, namely:

- *seq_goto*,
- *seq_prev*,
- *seq_next*, and
- *page_close*.

event_source	event_type	event
server	/courses/edX/DemoX/Demo_Course/courseware/graded_interactions/graded_simulations/	{"POST": {}, "GET": {}}
browser	page_close	
server	/jsi18n/	{"POST": {}, "GET": {}}
browser	seq_goto	{"old":1,"new":4,"id":"i4x://edX/DemoX/sequential/grade
server	/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_sequential;_graded_simulations/handler/xmodule_handler/goto_position	{"POST": {"position": ["4"]}, "GET": {}}
server	/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_problem;_free_form_simulation/handler/xmodule_handler/problem_get	{"POST": {}, "GET": {}}
server	/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_problem;_logic_gate_problem/handler/xmodule_handler/problem_get	{"POST": {}, "GET": {}}
server	/courses/edX/DemoX/Demo_Course/xblock/i4x:;_;_edX;_DemoX;_sequential;_graded_simulations/handler/xmodule_handler/goto_position	{"POST": {"position": ["2"]}, "GET": {}}
browser	seq_goto	{"old":4,"new":2,"id":"i4x://edX/DemoX/sequential/grade

Figure 4.14: Sample of problem events (MySQL Workbench screenshot).

The database query filtering options are four:

1. the user name,
2. the course identifier in the format of figure 4.14 in the *event_type* column,
3. the *problem_get* string appearing in that same column entry, and
4. the navigational events listed above appearing in that column.

The logic among them would be: 1 AND ((2 AND 3) OR 4). A check for null results is added prior to processing. All the events prior to *problem_get* are eliminated from the query as it indicates the start for the time measuring. The list should be cleaned so that it consists of the repetition of a *problem_get* and a navigational event subsequently, that is a starting and ending point pairs one after the other.

It was detected empirically by examining the MySQL *TrackingLog* table for problem events that it is possible that the navigational event that closes a problem never occurs due most likely to the fact that the system was not properly turned down, e.g. for operative system crashes, power outage and the likes. A threshold is then necessary to limit the error in the measure for this cases.

Finally, traversing the list of events, dates are obtained and their corresponding time for the problem accumulated.

This is the equivalent for problems of the *find_video_intervals* method.

time_on_problems

Argument: a list of *UserTimeOnProblems* objects (see table 4.15).

Returns: 1) a list of Python *datetime* objects with the days on which there is work on problems and 2) a list of the total time (in seconds) spent on problems per each day of the fist list.

It de-serializes the necessary information from the *UserTimeOnProblems* list and then computes the daily accumulates.

video_consumption

Arguments: 1) a *UserVideoIntervals* object (see table 4.15) and 2) a list of video durations.

Returns: 1) list of video percentages viewed and 2) list of total time spent on each video.

This method determines the time a user has watched a video without overlapping to know the percentage viewed, together with the total time of video watched including repetitions.

problem_consumption

Argument: a *UserTimeOnProblems* object (see table 4.15).

Returns: list of time spent on each course problem.

This is the equivalent for problems of *video_consumption*, but much simpler. It may seem logical to use a single method regardless of the type of resource to just determine the time spent on it by a user from a list of time intervals. However, the video context is different to problems as for video interval repetition must be distinguished while for problems there is nothing more than the simple time spent.

histogram_from_intervals

Arguments: 1) a list of interval starting points, 2) a list of interval ending points and 3) a list of durations for the videos in the course.

Returns: 1) a list of interval delimitation points (histogram bins) and 2) a list of the repetitions for each interval (histogram bins' height).

From the interval lists information to build the video repetitions visualization is computed. This visualization will look like a histogram. The two interval lists are merged and the starting and ending video points are added, that is, zero and video length respectively. This information is incorporated because the user may have neither watched the video from the beginning nor reached the end while playing. However, in the visualization, the whole video duration is to appear. Therefore these points are required just in case they do not come in the interval lists.

Duplicate points are removed from the merged list and then the latter is sorted out ascendant. This is the interval delimitation list. To determine the interval repetitions the method *get_hist_height* is created. It will require a list of points belonging to each interval. The easiest way to build this list is by using the intervals' midpoints which are simply determine by the average between the outer points.

get_hist_height

Arguments: 1) a list of interval starting points, 2) a list of interval ending points and 3) a list of points belonging to each interval (a single point per interval).

Returns: a list of the repetitions for each interval.

Interval repetition is determined by noting that it is the difference between opened and closed intervals at the position of the point being considered.

sort_intervals

Arguments: 1) a list of interval starting points, 2) a list of interval ending points.

Returns: the two lists passed as argument sorted out ascendant.

In working with video intervals, the need of sorting the intervals out is recurrent. It involves few operations but to avoid repetition they are wrapped in this function.

Sorting lists in Python can be accomplished natively in two ways: 1) the built-in function *sorted()* and 2) the list type method *sort()*. The second one is slightly more efficient than the first one. However, *sort()* is invoked from a list object. Hence, it can only sort the list from which it is called. It does not create any new object, just put the list elements in order. We need to sort two lists using the first one as criterion for the elements in the second one to keep their position relative to those in the first list. Therefore, *sort()* is useless for our purpose. We have to use *sorted()*.

The desired sorting is accomplished by a combination of Python built-in methods *zip()* and *sorted()*. The result must be converted back to the list format.

daily_time_on_videos

Argument: a list of *UserTimeOnProblems* objects (see table 4.15.

Returns: 1) a list of Python *datetime* objects with the days on which video-watching activity appears and 2) a list of the total time (in seconds) spent watching video per each day of the fist list.

The function loops the extraction of the information it needs from the *UserTimeOnProblems* objects and passes it to the "videowise" method *daily_time_on_video*, checking that the information is not null, meaning unwatched video.

A list is made with the dates for all videos, duplicates are removed and then the list is sorted out. Looking for video time watched day by day in the cumulative information obtained with *daily_time_on_video* method, a parallel list is created with daily time in relative position to the day list.

daily_time_on_video

Arguments: 1) a list of interval starting points, 2) a list of interval ending points, 3) a list of Python *datetime* objects linked to the interval starting points and 4) the same for the interval ending points.

Returns: 1) a list of Python *datetime* objects with the days on which video-watching activity appears and 2) a list of the total time (in seconds) spent watching video per each day of the fist list.

This function was created to obtain the daily time spent on a video for a the user. It assumes that the information passed in the argument is related to video and belonging to a single user as this is its role in the application. However, it can be perfectly used for content other than video and related to more than one user depending on the arguments passed in.

It controls the case in which the video has not been watched yet and the appropriate return values but leaves the core processing to the *get_daily_time* method.

get_daily_time

Argument and return values: idem to *daily_time_on_video*.

This method returns daily time devoted to the activity described by the arguments. Intervals start and end are both relative to video position (could also represent another resource but we use it for video). Times start and end are both lists of Django's *DateTimeField*.

Procedure controls the interval belonging to the same day to sum them up and whether the interval belongs to a single day or the interval outer points lay on different days, e.g. starting a day at 23:50 and ending the next day at 0:10.

datelist_to_isoformat

Argument: a list of dates in the format of Python's *datetime* objets.

Returns: a list of strings representing the dates in ISO 8601 format, "YYYY-MM-DD".

This simple function uses the isoformat() method of Python's *date* and *datetime* objects and loops over the list.

class_time_on

Arguments: 1) a list of days where at least somebody in the course has accessed a module type and 2) the time accumulated relative to each day in the first list.

Returns: 1) a list of Python *datetime* objects with the days on which activity on the module is registered and 2) a list of the total time (in seconds) working with that module per each day of the fist list.

This method computes the aggregated time (in seconds) that all students in a course (the whole class) have dedicated to a module type on a daily basis. It also controls the response in case of null lists. For the application this function is used both for problem and video time aggregates.

get_video_events

Arguments: 1) a student and 2) a video module identifiers.

Returns: 1) a list of five lists, one per video event type. Each sub-list will contain the video positions in which the event it represent occurred.

This functions queries the MySQL table of the *TrackingLog* model (see figure 4.5). It looks for those entries whose:

- *username* field matches the student identifier;
- *event* field contain the video module identifier using the minus sign separated serialization that can be obtained with the *html_id* method of BlockUsageLocator objects (this is an expensive although inevitable database operation); and

- *event_type* field is any of the events: *play_video*, *pause_video*, *speed_change_video* or *seek_video*;

Seek events will be split into two lists as they involve both an old and a new positions.

Depending on the event type the information will come in several ways. The event-distinguishing capability to get the video position will be left to another method, *get_current_time*. With the information it returns the different event lists are built.

get_current_time

Argument: an entry of a Django QuerySet for the *TrackingLog* table representing a video event.

Returns: the video position in which the event occurred.

The *event* field of the query is converted to a Python dictionary and the key to be inspected depends on event type as indicated in table 4.19. Its value will be the position relative to video length where the event took place. Notice that play and pause events key is very similar to that of speed change event, but not identical. They are easily mistaken and care should be put to avoid an error. For the seek event we have to retrieve two values for starting and end video playhead positions respectively.

Event(s)	Position key(s)
play, pause	'currentTime'
speed change	'current_time'
seek	'old_time', 'new_time'

Table 4.19: Event dictionary keys for video position.

4.16 Data querying functions

In this section we will examine those functions that will be used to retrieve data already processed from the database. In some cases, a slightly further processing will be still necessary to meet the presentation format. As a protocol between client and server sides of our application, empty values for the visualization data arrays with be set to *None* in Python (sever side) that will be translated to null in JavaScript (client side) at the time of encoding data with JSON.

get_module_consumption

Arguments: 1) a user name, 2) a course identifier and 3) a string describing the module type ('video', 'problem').

Returns: 1) a list of module names, 2) a list of total time per module and 3) a list of percentages of video watched for video only or a null list in case of problems.

This function will help us query the video and problem consumption in terms of time. It looks for those entries of *ConsumptionModule* table (see figure 4.12) whose *student*, *course_key* and *module_type* fields match the arguments respectively. Then it uses the fields *display_name*, *total_time* and *percent_viewed* (for video only) to construct the return list.

ready_for_arraytodatatable

Arguments: 1) a list of strings representing column headers or series names for the tables to be used by the visualizations, and 2) a variable number of arguments, that are lists containing column or series values for the visualizations.

Returns: an array in JSON format.

This method builds a JSON document ready to be used by the *arrayToDataTable* method of Google Charts library. With the information we query related to processed data, it adapts the content to an input understandable by the visualizations, that is also the most efficient way of passing the data in as we discussed in section 4.10.

get_video_events_info

Arguments: 1) a user name, and 2) a video identifier.

Returns: an array in JSON format with the event positions list.

This is the function in charge of getting the processed data for the Video events dispersion within video length visualization. Data here answers the question: At what time the user did what along the video? It queries the VideoEvents table (see figure 4.12).

Let us recall that the aggregate values are not stored for this visualization as they are simply the union of the events for every of the individual students. Therefore, if the function is requested the class aggregate, that is, data for student '#average', it will simply query for the video identifier. Otherwise, it will also query the database for a match with the user name.

Event list retrieved from the database are decoded from JSON, put together in the correct format ready to be used by the charts API and encoded back to JSON.

get_user_video_intervals

Arguments: 1) a user name, and 2) a video identifier.

Returns: an array with the video intervals and their repetitions in JSON format.

This method is charged with querying the table *VideoIntervals* (see figure 4.12) for the processed data for the Repetitions per video intervals visualization. This data when visualized will answer the question: How many times which video intervals have been viewed?

It will control the *DoesNotExist* exception if there are no entries in the table for the video and user requested.

Video intervals are stored just like they were obtained with variable bin sizes (it would be rare to find a repetition pattern with homogeneous bin width). However, variable bin or column size was not an option in the chart gallery except if we use one-second resolution intervals. That way, all atomic divisions will the the same width and those belonging to the same interval will have also the same height. It does not make sense to store in the database this one-second resolution values as they will be largely redundant and affect performance. It was decided just to store the intervals like they are and using this function to interpolate to get the one-second resolution at the time of request for visualization.

get_daily_consumption

Arguments: 1) a user name, 2) a course identifier and 3) a string describing the module type ('video'or 'problem'for the time being, although it will work properly for other modules provided that there are entries in the database for those).

Returns: 1) a list of days and 2) a list of daily time on the module type passed as argument for the days in the first list.

This routine queries the *DailyConsumption* table (see figure 4.12) and retrieves the information to return from the entries matching the arguments criteria.

join_video_problem_time

Arguments: four lists containing respectively 1) video-watching days, 2) time on video per those days, 3) problem-attempting days and 4) time on problems per those days.

Returns: an array in JSON format with the video and problem daily information coupled.

Once we have daily time spent on video and problems, we need to put these informations together so that they can be jointly represented in a column chart. This function tailors the processed data in this sense to the expected format by the *ColumnChart* type in the Google Charts gallery.

Chapter 5

Visualizations

After the design and development is done we look at the resulting visualizations, a sample of them, some special cases and test tasks carried out to ensure they work as expected.

A commonality for all the visualizations are the options for students and teachers. Students will be able to visualize their own information. Therefore, students visualizations have no student selector. However, they do have a video selector for those visualizations showing a single video, to be able to switch the video whose data is displayed. These visualizations are: "Repetitions per video interval" and "Video events distribution within video length". Teachers can visualize the charts for single students, any of the course enrollees. Thus why they have a student selector for each visualization, which enable them to select individual students as well as aggregates such as the student average, typically. They also have the video selectors for the single video visualizations in exactly the same way as students do.

5.1 Interface without data

The first case we examine is that in which there is no data to display for the current selection. An example for this case is shown in figure 5.1. Notice the user here is *staff*, the one with staff access among the four dummy accounts coming with the developer stack to assist with development and tests (see table 4.1). Hence, a student selector is available for the teacher in each and every one of the visualizations. The three dummy student accounts are used in the different visualizations: *audit*, *verified* and *honor*. They are selected on purpose together with video to illustrate the case in which there is no data for any of the visualizations.

The equivalent situation for a student is as shown in figure 5.2. There, no student selector appears as the privacy policy prevent them to access data concerning their partners. They can only select among videos in the visualizations with that feature.

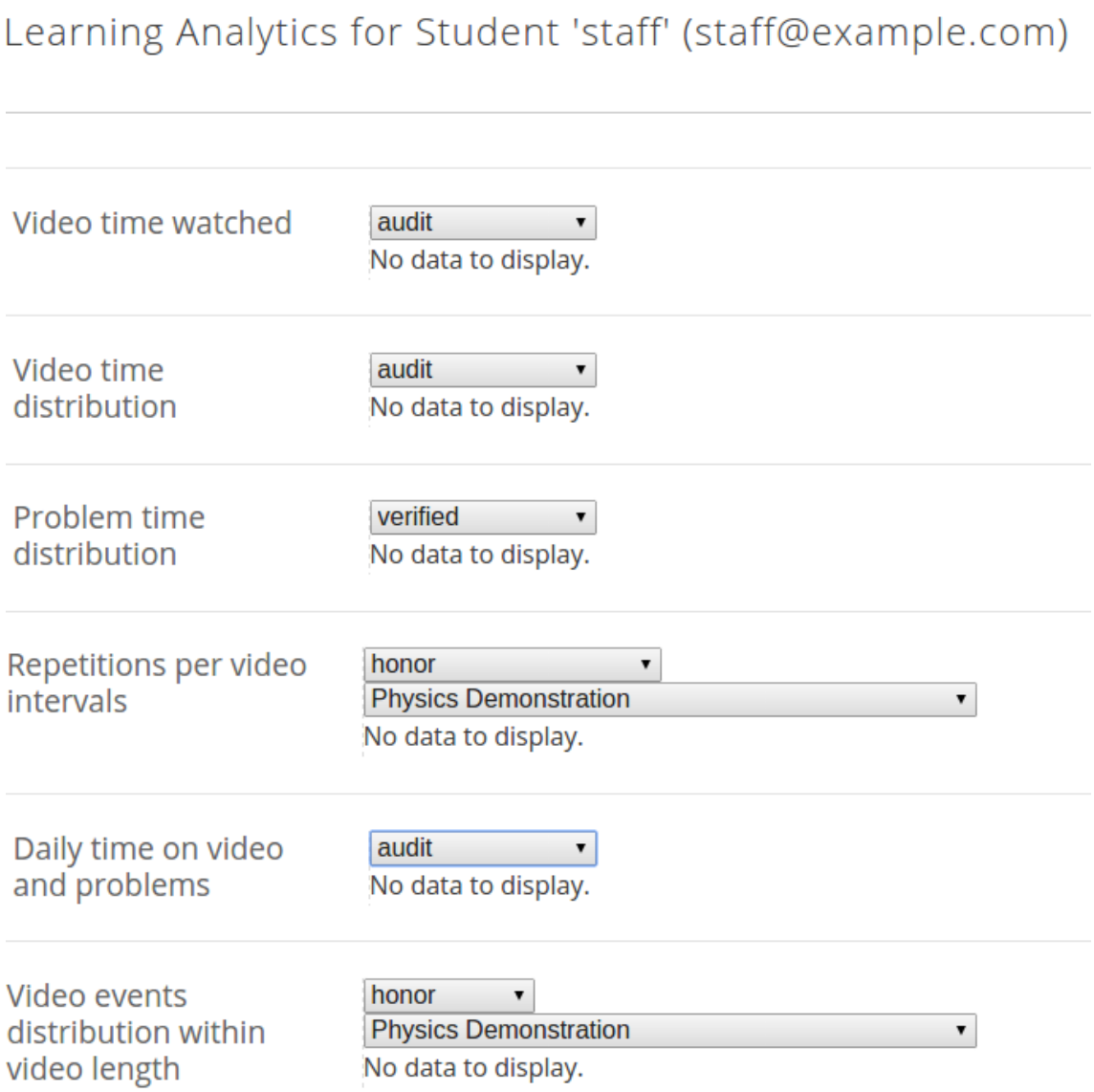

Figure 5.1: Course staff interface screenshot. Case: no data.

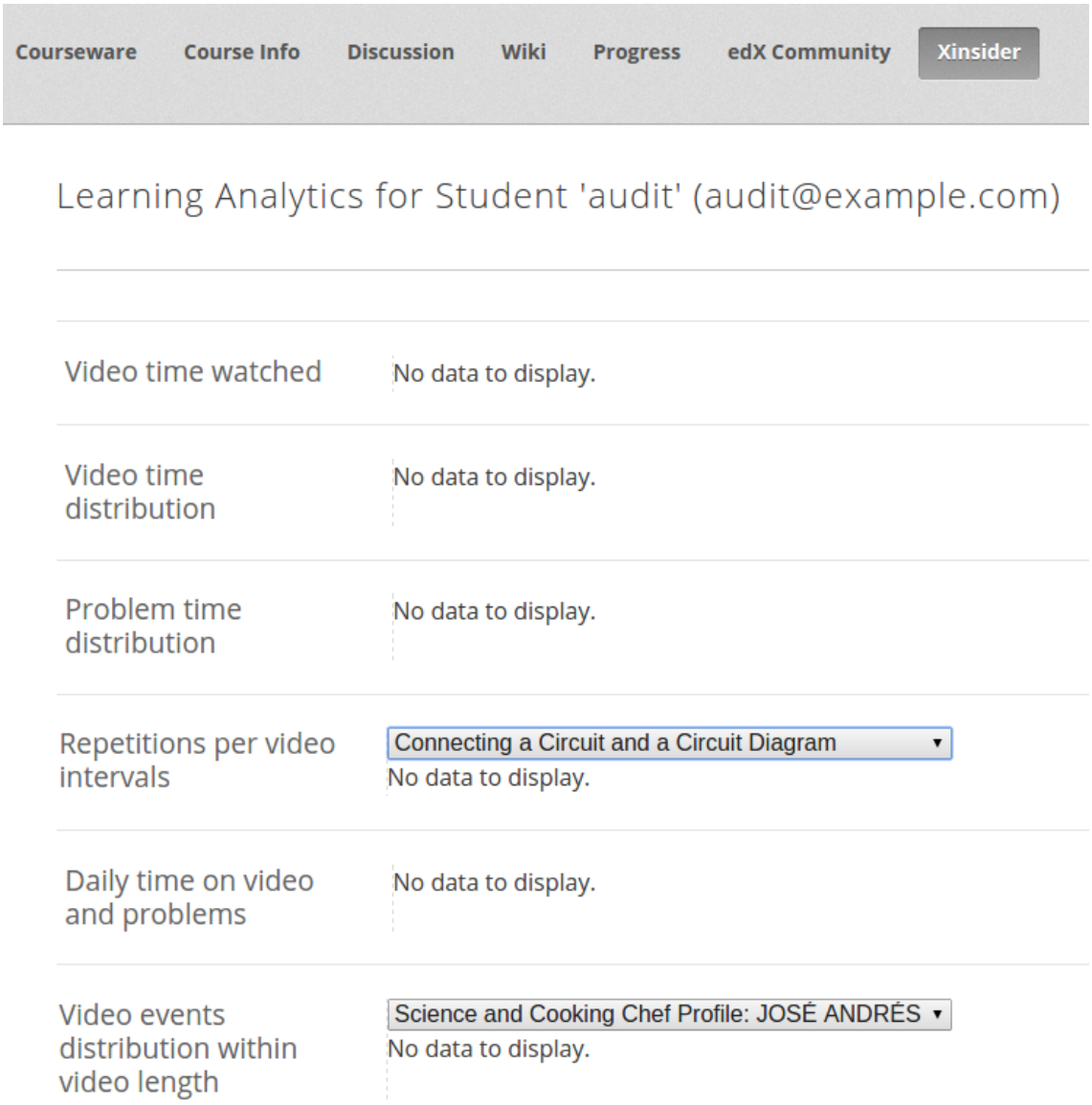

Figure 5.2: Student interface screenshot. Case: no data.

5.2 Video time watched

An example of the first visualization is reproduced in figure 5.3. There appears the visualization of the student average for a user with staff access, typically, the teacher. Videos with long names have no space for the whole name, but it can be inspected by mousing over the name on the horizontal axis or directly over the bars. This is a perk of Google Charts, some automatic annotations as we mouse over the different visualization parts to get highlights, clarification or precision at the time of reading the visualization. In the figure, the mouse is over the non-overlapped time spent on the video "Science and Cooking Chef Profile: JOSÉ ANDRÉS" by user *honor*. That allows us to know the exact percentage if we are interested in. The visualization also shows that that user has not watched the three last videos of the course yet, and that he/she tends to repeat fragments although they have not completed at all the fist two ones. A 24% of the first video has been watched and a 7% of the second. A possible interpretation of these bars is that the user may be facing problems understanding the videos. He/she has started both but finished none. The total time doubles the non-overlapped one as if the student were starting over both of them for some reason. This may raise questions as: is the video clear and properly levelled?; is it good-enough to capture the students attention?. The much higher bar for total time may be indicating effort or struggling on the student side. Yet these are results for a single student. It would be helpful to see how what the visualization is showing compares to the class aggregate or to other students to check if the hypothesis formulated hold.

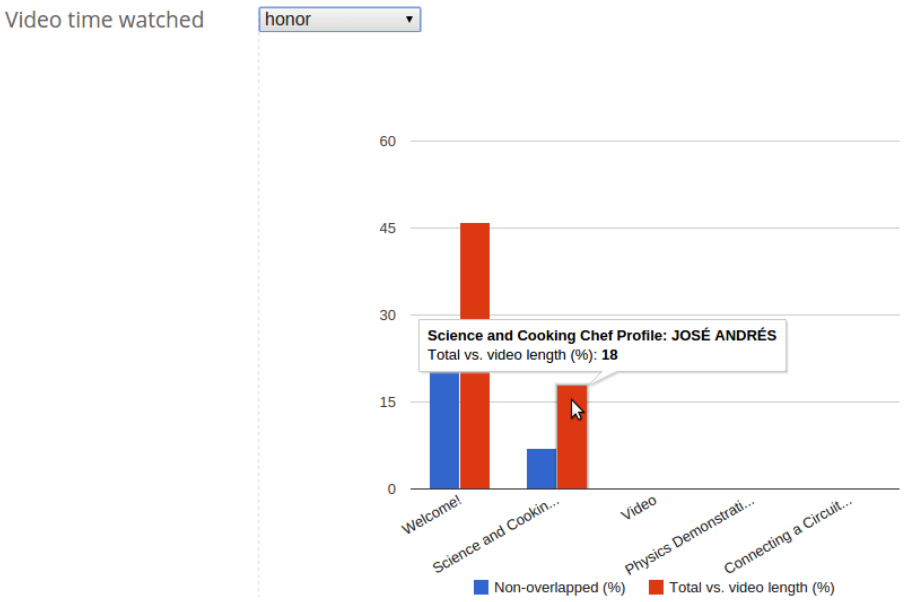

Figure 5.3: Example of video time watched visualization

Another example for the same visualization is reproduced in figure 5.4. The image was captured before selectors were implemented and functions to obtain students aggregate were defined. It was even done with a previous version of the development stack. Notice the video names and quantity differs in some cases with respect to figure 5.3. Although the development stack comes with the "DemoX edX Demonstration Course" its content slightly vary among versions, especially the videos that are included. On the contrary, course structure remains the same. Notice between the figures that some videos differ.

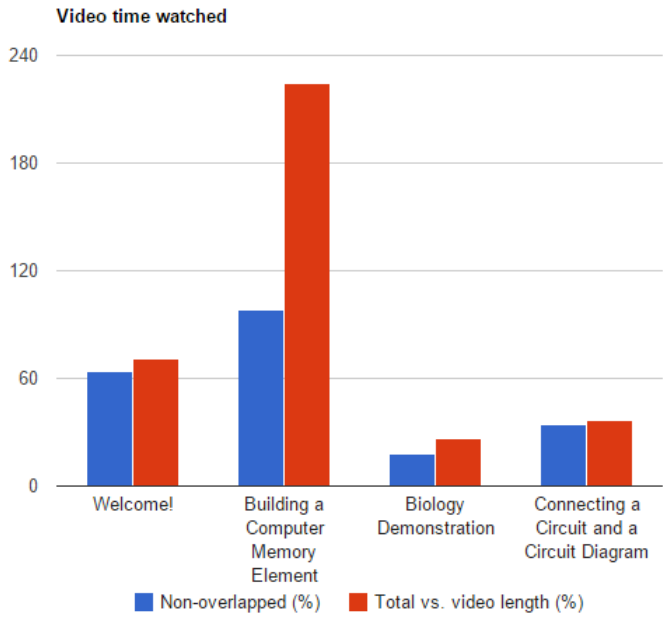

Figure 5.4: Another example for the video time watched visualization

In the visualization example of figure 5.4 the two time measures are very similar except for the video "Building a Computer Memory Element". There the video was completely viewed and the total time the user spent on it was twice the video length. A possible hypotheses is that this was the video that captivated the student the most as it was the only video he/she viewed completely and where they spent most of their video time with great relative difference with respect to the remaining videos.

5.3 Video time distribution

By the way, the video time distribution is better seen in the second visualization. Figure 5.5 shows an example of the slices of this distribution for the student average. In it, video "Science and Cooking Chef Profile: JOSÉ ANDRÉS" dominates the scene followed by "Physics Demonstration". This visualization is built from

time data expressed in seconds that is automatically converted to percentages. The Google Charts annotations available for this kind of visualization allows us to know both values by making a mouse-over movement. In the example, for the video "Video" it reads that students spent here an average of six seconds which account for nearly a six percent of the time they devoted to watch videos. Let us recall that this visualization does not weight video time with respect to video length. Therefore, the time spent on a video may be bigger due precisely to the fact that it lasted longer.

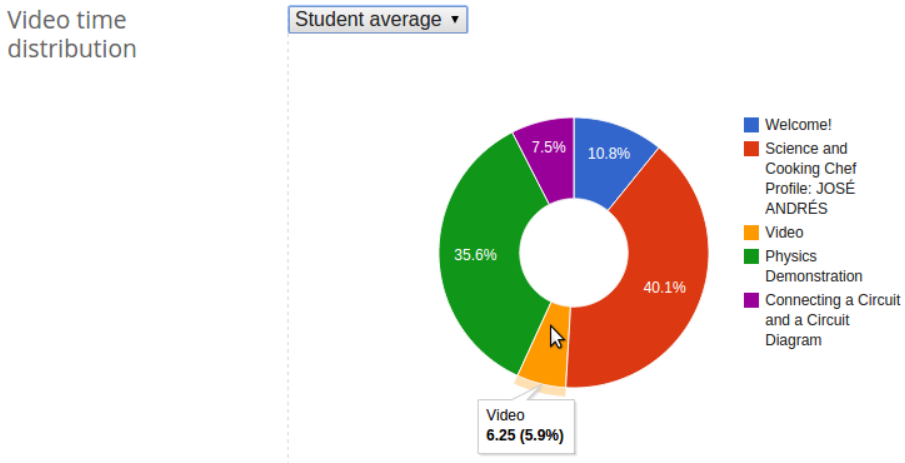

Figure 5.5: Example of video time distribution visualization

5.4 Problem time distribution

There is an example of the problem time distribution in figure 5.6 for an old platform version. There were 21 problems in that demonstration course, of which the user interacted in this case with only three of them. It is worth to point that although the names look like problem types, they are individual problems named after their type to present the user with the different problem features and interactive experiences available at edX.

Another example for this visualization with selector implemented is on figure 5.7. The class average is selected. It shows students have interacted with five problems of the course. The mouse-over annotation tell us similar information to that of the version of this visualization equivalent for video, as the chart types coincide. For the example, to the problem "Mathematical Expressions" have been devoted seven seconds in average that accounts for the 15% of the time students have devoted to problems. Notice also that in the legend the fourth problem appears with no name. That is because it does not have a name indeed. As a result, we do not

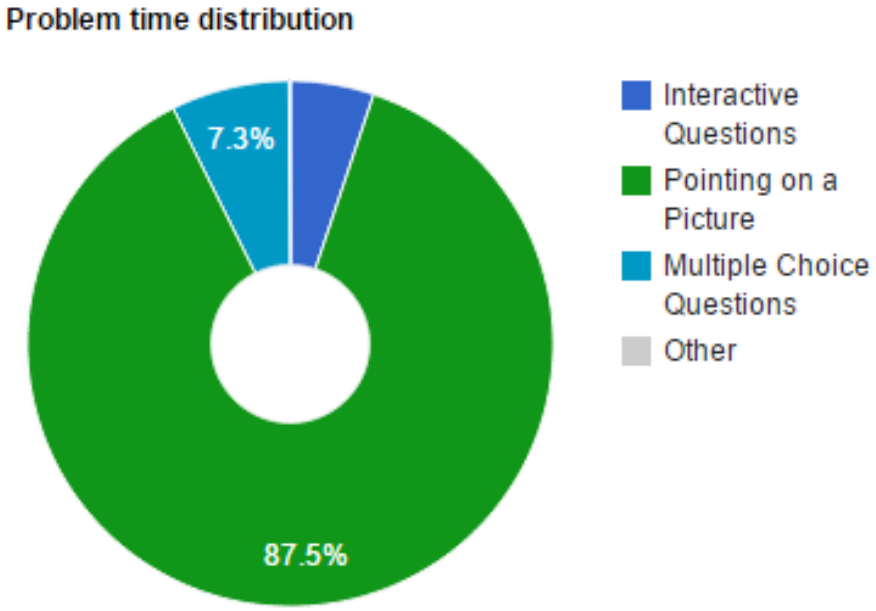

Figure 5.6: Example of problem time distribution visualization

know exactly to what problem students dedicated almost a quarter (24.1%) of their time on problems. We could have chosen an alternative name for example with the patter "Chapter X. Problem Y.", where X would be the number of the chapter inside the course, and Y the relative number of the problem within the chapter. We think, however, this name is not meaningful and does not worth it. It is strongly recommended to teachers that they name all the problems and course resources in general so that the most semantic possible analytics are presented.

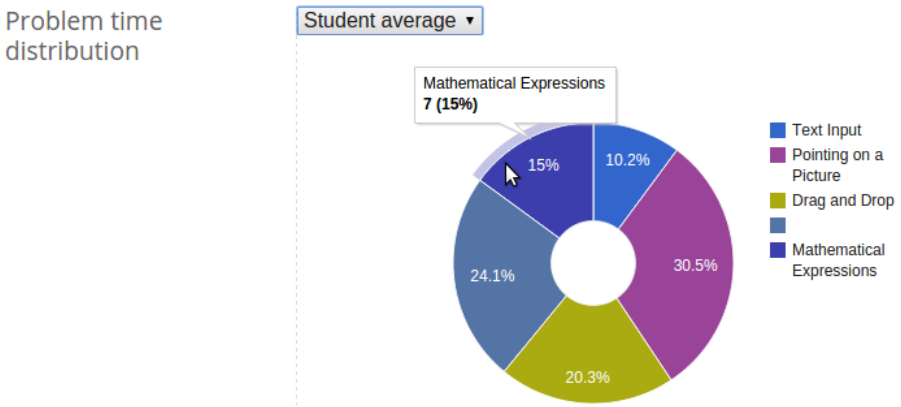

Figure 5.7: Another example of problem time distribution visualization

5.5 Repetitions per video interval

Moving on to the video repetition visualization, figure 5.8 reproduces an example of the video repetition profile of a student for the video "Building a Computer Memory Element" with duration 131 s (see table 4.16). Along the horizontal axis we can see the video length. The profile shows there was an interval watched five times, that most of the video was viewed four times and the end a single time. The video was completely watches. This is an example for an old development platform version and lack still the selectors.

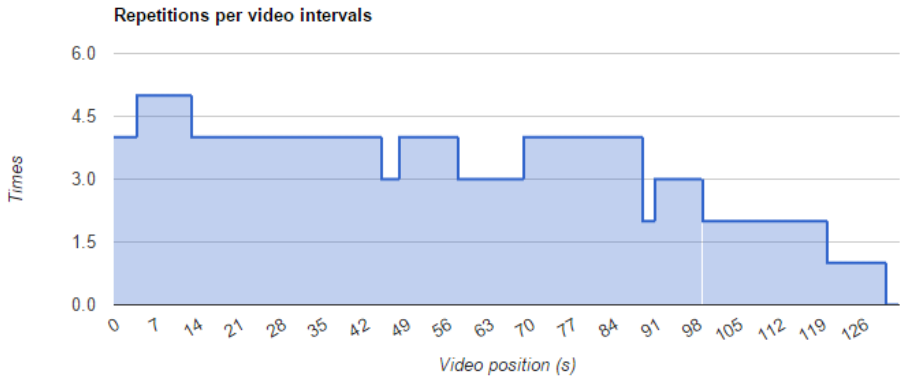

Figure 5.8: Example of video repetitions visualization: irregular profile.

Example in figure 5.9 on its side shows a much more regular pattern, in this case, for the user *verified* when watching the "Physics Demonstration" video. A hypotheses we could pose is that for some reason the user has skipped the first part of the video and started to view it from the second 48 up until the end. Perhaps due to the recommendation of a colleague. This may raise the questions: Do the first 48 seconds of the video lack interest?; if yes, why?.

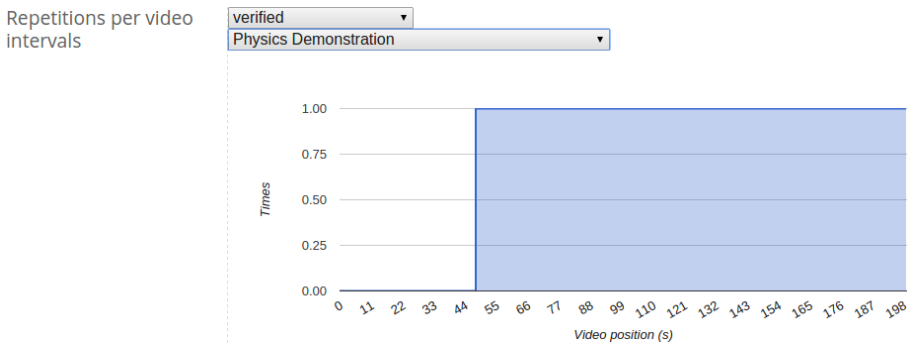

Figure 5.9: Example of video repetitions visualization: regular profile.

As one of the proposed chart types for this visualization discussed in section 4.9, we were experimenting with the timeline type. Although not completely debugged, figure 5.10 is a screenshot of the visualization we were working on. It was finally discarded in favor of the stepped chart that better showed the profile and made it easier to detect video playing peaks and sinks in intervals. The timeline approach consists of a row for each video named after the video itself. The timeline extends up to video length. In the version of the "DemoX edX Demonstration Course" from which the video come, there are four videos with different length as is reflected in the bar's extension. Each bar is segmented per interval and intervals are labeled with the number of times it has been seen. An annotation is prompted when the user mouses over the intervals, indicating the number of repetitions at the top, video name and interval range in the center and the interval duration. In the example, the time axis, i.e. horizontal, is not properly configured and therefore the annotation reads: "0.184s - 0.189s" and "0.005 seconds" instead of the correct values "184s - 189s" and "5 seconds" respectively. Peak detection is harder with this visualization as the profile is flat and the number should be considered, with the added difficulty that for small intervals the number is omitted as can be appreciated for a number of intervals in the visualization, making it necessary to mouse over all of this intervals. As a benefit, however, it enables the user to have an idea of all videos at a glance, as with the stepped area chart a single video is shown at a time.

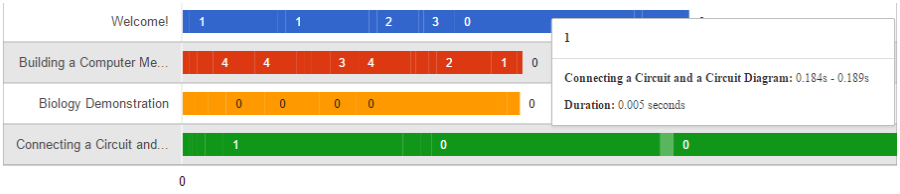

Figure 5.10: Discarded timeline chart for video repetitions visualization.

5.6 Daily time on video an problems

Daily time on video an problems visualization require interaction for a relatively long period of time to draw reliable conclusions. This is not the case of the incipient interaction generated for user *honor* to show the example in figure 5.11. It does not affect the purpose, however, of illustrating how the visualization is intended to look like. The visualization shows the expected pattern of the user spending more time solving problems than watching videos as the former implies active learning by doing things and putting hands on to the taught topics.

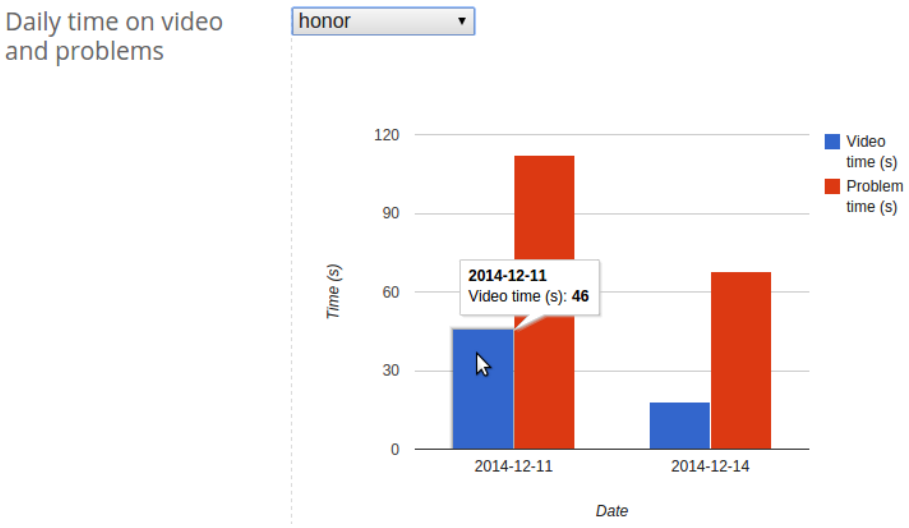

Figure 5.11: Example for the daily time visualization

5.7 Video events distribution within video length

Finally, the aggregate of interactions for all the students with the video "Science and Cooking Chef Profile: JOSÉ ANDRÉS" is shown through the events visualization in figure 5.12. The amount of test data is not big-enough to formulate hypothesis. The number of *seek from* and *seek to* events must coincide as they are generated from a single user event when changing the video position by means of any of the available ways to do this. It can be seen on the visualization, however, that there are six *seek to* events but five *seek from* ones. This is because there are two of the latter superimposed at the starting position. This suggests that there should be an indication of the even number, perhaps by modifying the size of the points.

So far in this chapter we have seen a number of samples for the implemented visualizations and possible dummy interpretations as the data is generated for test. They have to be applied to real courses and learning environments to get the fullest out of the visualizations for both students and teachers.

In the following section of the chapter we examine some of the tests manually performed on the visualizations.

5.8 Tests

Here we will list some high-level tests for the visualization. No automatic tests were implemented and the list is in no case thorough.

- Test the development in several browsers.

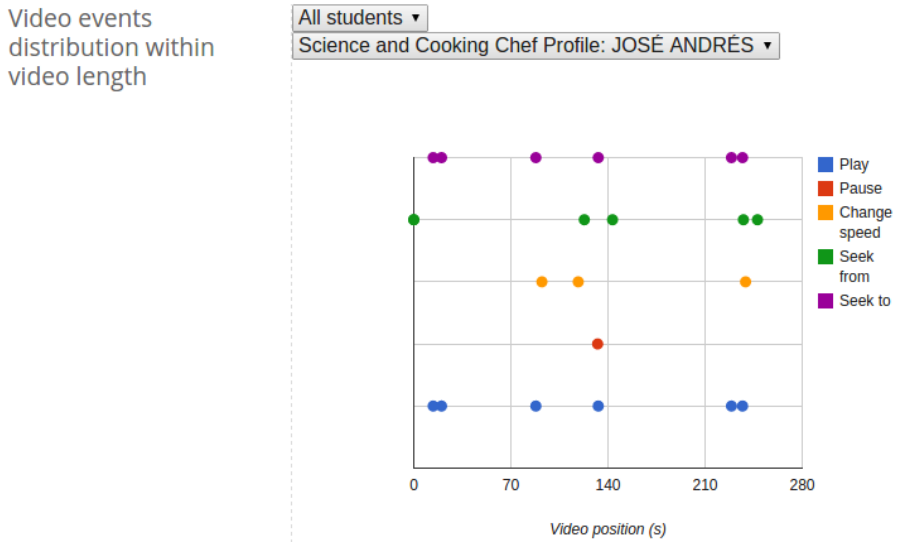

Figure 5.12: Example of video events visualization

- Check that the "No data to display" message works correctly for empty datasets. Ensure using MySQL Workbench that this is consistent with database records.
- The total video time watched must be greater than or equal to the time taking only into account non-overlapped intervals. Figure 5.13 shows an inconsistency in that sense. There, for the fifth video, "Connecting a Circuit and a Circuit Diagram", the percentage without video overlapping surpasses the total one, which makes no sense.
- Non-repeated video percentage should not surpass the 100%.
- Verify the AJAX request is working properly using the console and Developer Tools in general of the browser.
- Examine all the four students and aggregates for all the six visualizations.
- Check non-English characters are displayed correctly. For example: the name of the video "Science and Cooking Chef Profile: JOSÉ ANDRÉS". Some encoding adjustments were needed.
- Verify total time bar values coincide with the entry time for the donut chart of video time distribution using the Google Charts annotations feature.
- Contrast for consistency *Video time watched* visualization and that of *Repetitions per video intervals*. Inconsistent example: that the former shows the whole video was viewed but in the latter there are intervals that appear unwatched.
- Check that the aggregates are consistent with the individual student values for all the six visualizations. An example of inconsistency is shown in figure 5.14. There, it was noticed an excessively high number of repetitions at the end of the "Welcome!" video (randomly used for test, it could have been

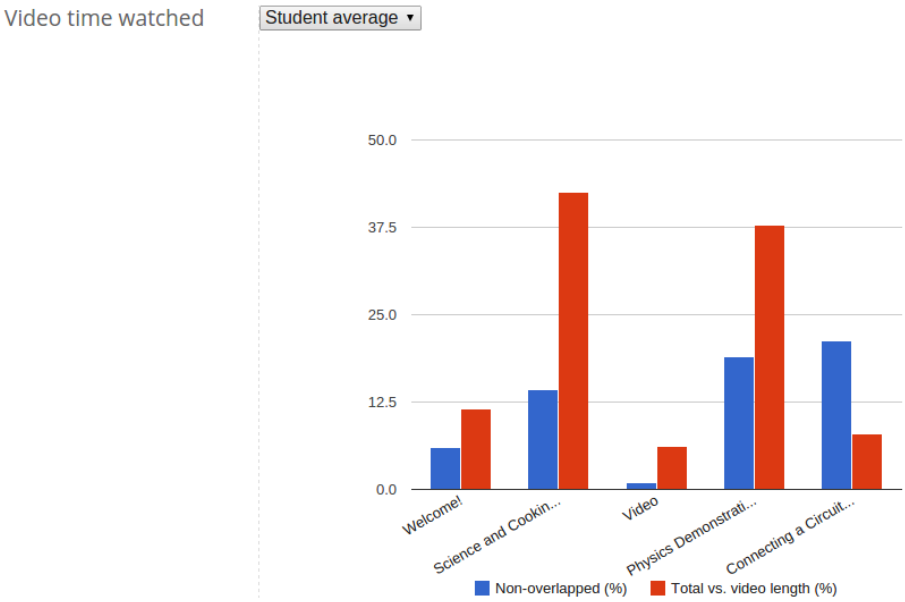

Figure 5.13: Inconsistency example between video percentages.

any of the videos in the course) that didn't seem to match the sum of the individual student repetitions. There was a null video interaction for one out of the four dummy student accounts in the developer stack. For the three users with activity (generated for test), their profile for this video were was put one below the other together with the aggregate. The visualizations were aligned to make a vertical correspondence of the time instants with the use of a vertical bar. With this bar located at the last interval approximately, the error in the aggregate was clear. Students accounted for $3 + 2 + 1 = 6$ repetitions respectively from top to down. However, the sum results in 19 repetitions in the aggregates. To trace the error, the lists for the raw intervals, the interval division and the interval high were carefully examined. It was found that the aggregate variable was not properly re-initialized after the loop for each interval.

- Consistency between aggregates for the only visualization with two of them, the intervals visualization. The times without limitation should be equal or greater than those limiting the students contribution to a single one.
- Verify dates on the horizontal axis in the daily time visualization appear in chronological order.
- Contrast the consistency of the video events visualization with the other video-related. Inconsistency example: there is no play event for a video but some watched time for the same video.
- Try to access the visualizations' page without logging in to test the prevention of unwanted access is working properly.
- Manual calculations of simple data to check if the algorithms work as expected.

- Interact with the platform in a way that can be easily distinguished in the charts, execute the data processing and verify the charts are update according to the interaction.

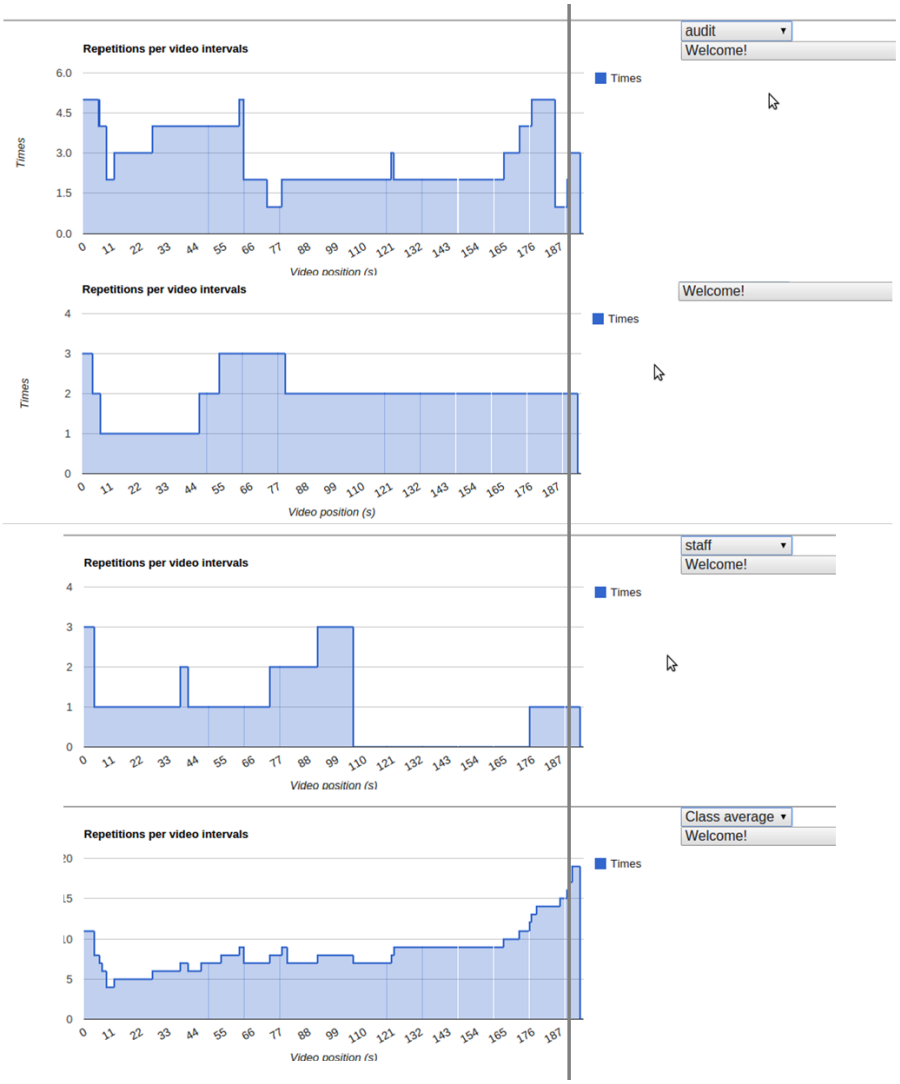

Figure 5.14: Inconsistency between individual and aggregate values for the video repetitions visualization.

Chapter 6

Conclusions and future work

In this chapter we wrap-up the project work, present the conclusions we arrived at and what has the project meant. We will also indicate some proposals to continue working in this line.

6.1 Conclusions

In this project we have extended the learning analytics support of the Open edX platform by designing and implementing a set of six new visualizations mostly related to the analysis and/or inference of student characteristics stemming from a time distribution approach, in opposition to the more traditional analysis oriented to marks and results. The single learning analytics visualization at edX for students is a progress column graph. For teachers there are also demographics, course enrollment and student activity information. The visualizations we have created for students and teachers as well are: a column chart for video time watched per video in all and without overlapping, two pie charts for video and problem time distribution respectively per course, a stepped area chart for the video repetitions profile, a column chart for the daily time students have spent on video and problems and a scatter chart to show where and which video events have been triggered by the user when watching every video.

For that purpose, familiarization with technologies new to the author was a must. Vagrant fundamentals was a start to understand the replication of the development environment and to use it as a tool to debug installation errors that can be considered the first hurdle and entry barrier to the project. We discovered with Django what a web framework is and learned a new programming language, Python, upon which Django is built. This was perhaps the easier part due to the similarity of structures with a number of programming languages we have used throughout different subjects and laboratory assignments. We found the benefits of templates with the Mako templating language. In the way we exercised web technologies such as HTML, JavaScript and CSS at the time of discovering the powerful AJAX technique for devising dynamic and fast web pages, as well as the JavaScript-based Google Charts

library. We learned that relational databases were not the only type at all, but there were non-relational ones of which MongoDB was our ally on getting to know these databases. We saw that they do have differences, but that they also share common or equivalent terms or concepts.

The confluence of so many technologies and techniques made this project daunting, yet an opportunity to work on a real life application with a solution available through the community to anybody willing to include our module in his/her edX instance.

The most challenging task was to understand the architecture of the Open edX Platform, the what, how, where and why behind the code. What is done, how is done, where is the code that does it and why is that done are the guiding lines. The documentation fell short of thorough to start developing and prone to be obsolete due to constant platform updates of different scope. Expressions like: to do, to be deprecated, obsolete and similar were found often in the documentation as well as in the code. It is a live platform open to the community with a growing number of contributors. So it is far from static, which is good on the other side as it means the platform is always getting better and better, new functionalities and improvements are often added.

Therefore, the best documentation was the code itself. Reading code is a hard, time-consuming task but it is also a great way to learn and develop analysis skills. It was of extreme importance to get a good understanding of the code. We were not developing an application from scratch, but willing to make a contribution to an existing, awesome and exciting project. Hence we needed to work with the platform convention, Python guidelines, and most important, keep faithful to the DRY (Do Not Repeat Yourself) principle. The latter implies that nothing was created or defined prior a code research to guarantee we were not duplicating functions or tasks, what would result in our work not really adding value. We are not sure if we fulfilled the DRY principle up to the last line as there is a bulk amount of code, but we tried hard to keep attached to this principle.

Trial and error became customary in the project development. Once again, we experienced first-hand that debugging is more than a half of a software project development.

We saw in the firsts chapters that the willingness of improving the educational process is not new, but today there are more tools to do pedagogical research than ever before and what matters the most: data. Online platforms offer a bulk of data that can shed light on the learning process, what learners have in common and what is specific to a group of them. Data to decide on hypothesis of historic learning theories or to devise new hypothesis or even theories. This project is a small example and modest contribution to unlocking the potential of data for education, personal reflection and perhaps the most relevant outcome: decision-making support.

We did not know in the beginning that we were going to work with the Open edX platform, we have not even heard of the platform at that time. Our contribution platform was first Google Course Builder, also open but discontinued in favour of

edX. We wanted though to work on the learning analytics fields and we have kept faithful to that mission.

As part of the project a paper was written for and included in the proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'14) held in Salamanca, Spain on October 1 to 3, 2014. It was an opportunity to showcase our line of work that at that time were still in an early stages, within the track 5 of the conference: "Dealing with complexity: Educational data and tools for Learning Analytics". The rationale of the track stated[1]

> In the last two years, Learning Analytics is undoubtedly among the most trending topics in Education. Since there is currently not one, but multiple approaches to the topic âand even definitions of Learning Analyticsâ the range of different studies, experiences, tools and applications for Learning Analytics is growing at an overwhelming pace.

With this vision, the track pretended to be "a facilitator to share experiences with the creation and application of tools for learning analytics". There we shared our experience with edX in the paper "Towards the development of a learning analytics extension in Open edX" [44], whose abstract is reproduced below:

> The emergence of platforms to support MOOCs (Massive Open Online Courses) strengthens the need of a powerful learning analytics support since teachers cannot be aware of so many students. However, the learning analytics support in MOOC platforms is in an early stage nowadays. The edX platform, one of the most important MOOC platforms, has few learning analytics functionalities at present. In this paper, we analyze the learning analytics support given by the edX platform, and the main initiatives to implement learning analytics in edX. We also present our initial steps to implement a learning analytics extension in edX. We review technical aspects, difficulties, solutions, the architecture and the different elements involved. Finally, we present some new visualizations in the edX platform for teachers and students to help them understand the learning process.

For the author of this project it was a valuable experience to contribute to the paper together with the feedback received from the paper's co-authors and referees as well.

Finally, the most relevant landmark of the project is the release to the community. For that, this contribution together with other created in parallel by my colleague Javier Santofimia Ruiz, were integrated in the laboratory in collaboration with our colleague José A. Ruipérez Valiente. In the fusion as a single module, the following steps were taken:

- unification of the files where the Django project looks for the URLs,

[1] http://teemconference.eu/2014/tracks/learning-analytics/

- fusion of the setting files for the development stack,
- putting together the Mako templates,
- merging of the APIs created,
- debugging of namespaces incompatibilities,
- joining the tasks planned with Celery for both applications to be run in a sequence,
- unification of the Django views including the templates' context,
- harmonization of the AJAX requests, and of course,
- homogenization of user interface.

The changes in user graphical interface include:

- Learning Analytics tab in edX navigational menu;
- sub-navigational bar to group the visualizations of both projects in three categories: Video, Problem and Activity;
- changes in color palettes; and
- re-arrangements of titles, visualizations and legends positions.

The integrated contribution will be published as a Github fork of the master edX project. Thus, it will be open to edX instances administrators, developers and anybody in general to whom it may result of interest.

Although we are very proud of it, this landmark does not mean that the job is finished at all. It still needs to undergo more verification, battery tests in real learning environments and the ultimate phase of any software development, that is, maintenance.

In the next section we propose some future work on the application and discuss a number of improvement opportunities.

6.2 Future work

The module needs to be assessed with a set of tests mainly for teachers but also for students. The test battery should include usability, effectiveness and visualization interpretation assessments. It is necessary to gain feedback and insights from the real platform users what they find useful and what not. Polls are a valuable statistic tool in this sense. They will help determine if the visualizations are clear-enough and if the indicators they represent are correctly interpreted. Feedback is key for future improvements.

Xinsider fits into the edX platform, but as a Django application, it is portable and reusable. This benefit might be exploited to adapt the application for its integration into the Insights module, the edX's learning analytics development project based on streaming events with Django and Mongo technologies.

We saw in the introductory chapter how technologies of different times have found also their use and awaken expectations in education although this were not their

primary purpose. We talked about the motion picture, radio, television, computers and the virtual world that came with the web. What is the most trending device nowadays if not a mobile phone? People, especially young ones, carry their phones with them wherever they go. They love their cells and the things they can do with them. Applications for learning have thrived as for almost any other purpose. Tablets are currently being used to support the learning process in some schools, whose number is growing. The edX platform team is aware of the global trend of going mobile and that the future points in the mobile direction. In January 2015 they echoed in a social network (see figure 6.1 left) an article [24] of the "Inside Higher Education" portal highlighting: "Education for the masses [...] will take place largely on mobile devices". In the same blog post can be read: "Future is mobile learning", and the following forecast:

> The campus educational model, and the laptop (keyboard / screen) learning model, will not disappear. It will grow, and grow quickly throughout the emerging world. However, this growth in traditional models of postsecondary education in the newly emerging economies will be swamped by the new mobile and competency based models just beginning to emerge.

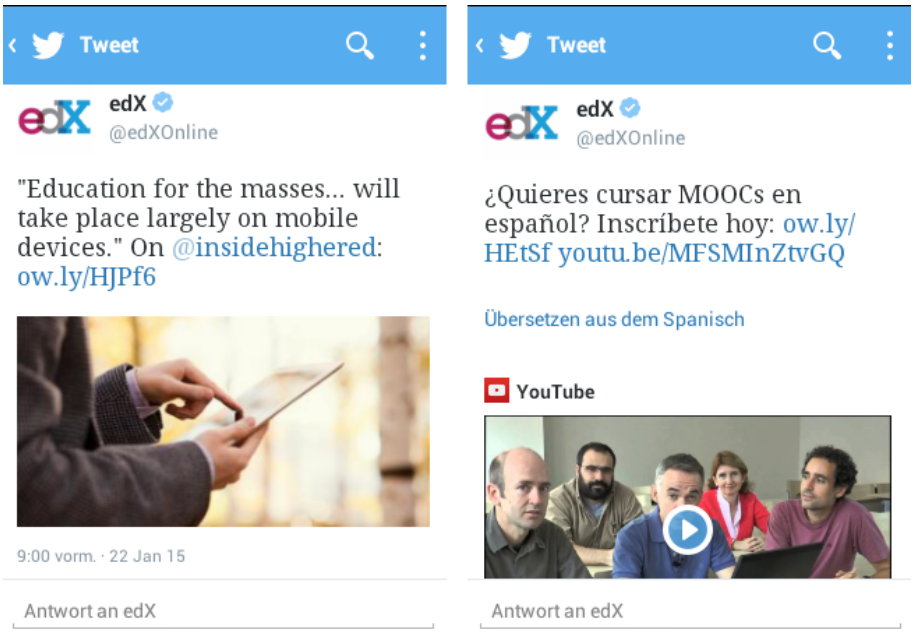

Figure 6.1: Two edX team posts on social networks (mobile screenshots).

With this big introductory reflection we intend to stress the importance of taking the next steps oriented to mobile devices and access. Hence, the most important future work in our opinion is to adapt the design to fit a mobile device and offer

tapping, appealing features proper of that devices. We think there is a lot of work to do in learning analytics in general but especially in mobile or tablets, by both using them to present the analytics but also to get data from them relevant to the learning process, and of course, with the user approval and the principle of anonymity. "Nobody" knows better a person than his/her mobile phone. Data in this sense is increasingly being used for commercial and advertising purposes to better target the audience of products and services. Learning analytics should not be foreign of the opportunities these devices offer.

Apart from this big, suggested line of work, it follows a number of to-dos and improvement opportunities to the application developed in this project.

The user interface is susceptible to improvement. Perhaps the visualizations may be grouped under a sub-navigational menu to avoid the several scrolls between the first and the bottommost visualization as well as facilitating the user to find what he/she is looking for. When it comes to integration with the edX environment it is key the use of the edX theme and color palette in the visualizations so that they do not look foreign to the platform, instead of using the Google Charts default color set.

Users with staff access are currently included in the class aggregates. As they are often teachers or researchers, they should be removed from the class aggregates in order that they do neither contribute nor impact the measure. It was included this way in the development to have a better sample for test and data sources as only four users are preloaded in the developer platform.

At this moment, student selection is "chartwise". That means if a teacher wants to examine the visualizations for a certain student, they have to do that one by one. It would be helpful to have a global selector for changing the student of all visualizations at a time.

We saw at some point that the table registering the events log had two columns for date and time and that these two columns differed with respect to each other due to time zone adjustments relative to the system. As this is an application destined to a worldwide public, the improvement of the time zone capabilities of the analytics module is key for the customization experience.

Google Charts were great for our purpose. Nonetheless, they are not the only charting library but there are tens of them. We made the decision based on previous works and the simplicity of the technology as well as cross-platform promises. It would be interesting however apply to the module other visualization tools to see what they have to offer. A systematic comparison for analytics could prove valuable and drive the creation of visualizations specifically oriented to this field.

Connectivity is one of the most beloved characteristics of the online world today. Users are happy to share their experiences through their social networks. This reality is not foreign to edX, which has an *edX Community* tab in the navigational menu for users to join the conversation on social networks and see what their counterparts are up to. It has been shown in several studies, including ones performed with data generated at edX courses, that social engagement is one of the keys if not the

key, to prevent high drop-out rates. All of this make the built-in social network integration in edX a rich data source to know about trending topics, what do the users think about newly released edX features or common ones, what do they think about the course resources, and so on. Analytics of social networks involve natural language processing techniques with the inherent difficulty coming from the variety of languages. English is the language for the vast majority of edX courses but new courses are increasingly created in other languages to overcome the language barrier entry to edX for those non-familiar with the English language or preferring to take courses in their mother tongue. As an example, figure 6.1 right shows an edX team post in a social network promoting Spanish-language MOOCs available at edX.

Apart from social network integration, edX features a discussion forum for students to communicate, interact, help each other, and so on. The use of natural language processing techniques to analize the discussion forums could make possible to detect trending topics, off-topic threads and the way in which this tool is used.

EdX has intrinsic localization features for users willing to contribute to the translation of the platform framework. In the Open EdX Platform page, it is stated that they "are particularly interested in people [...] who can assist with internationalization and localization". It is not the job of developers to translate the platform, but they can develop having in mind localization facilities that are already native to edX as well as others they devise for this purpose.

Video functionality is currently done for Youtube videos only. Video duration is the information that edX lacks and therefore it is obtained through the Youtube API. It would be useful to extend it to other video-platforms including alternative ways for course authors to enter the video duration explicitly at some field if they would like to use a video in a different platform.

There is a huge potential for analytics with the big number of events that remain without taking the fullest out of them. Many visualizations may be designed and implemented and high level parameters inferred from this rich source of data. Of course, a vast amount of visualizations may result in teachers or students getting lost in the middle of that bulk of information. The better architecture in our opinion, is to develop as much analytics visualizations as possible, but giving the user the choice of which are to be included. Implementation could be done by using a settings variable to control the visualizations to display, or plugin-like code additions or including in the page a multiple choice selector for the user to mark the visualizations of their interest.

The project visualizations that show time information currently use the second as unit. This was a development decision that allowed us to spend a lesser time generating events for test with a bigger resolution than if we were using minutes. Nevertheless, the time students devote to watch videos and solve problems is more likely to be in the minute domain. As the desired resolution may vary depending on user purpose and resource characteristics, the best solution would be to implement selectors to change the time unit depending on the resolution that best addresses the user needs. Sometimes it can be interesting to analyze progress up to the second

level but it may also be irrelevant in some cases where the time devoted in minutes or hours could make much more sense.

6.3 Epilogue

MOOCs are "*one* way of learning" [8, min. 1:00] in the own words of Dave Cormier, who created the term. Even if they are not or will not become *the* way of learning, they are worth to take advantage of as a way, a possibility, a resource in our best interest. They are worth to exploit, to pursue, to take the most out of them. New educational technologies will appear either in the near feature or in the long run for sure; one technology after the other; it has always been and always will be like that; but for the time being they are the "latest innovation in teaching" [34, p. 2]. They are not flawless, much like we are not either, but they are awesome and waiting for the students, the teachers, the researchers and the people of tomorrow. They are waiting for us.

Bibliography

[1] Amigot, M. (2014, October). *The ultimate guide to Open edX: A strategic guide to engage the most advanced technology for blended learning* (2nd ed.). New York: IBL Studios Education. Retrieved from http://iblstudios.com/edx-guide

[2] Atkins, D. E., Brown, J. S., & Hammond, A. L. (2007, February). *A review of the Open Educational Resources (OER) movement: Achievements, challenges, and new opportunities*. William and Flora Hewlett Foundation. Retrieved from http://www.hewlett.org/uploads/files/ReviewoftheOERMovement.pdf

[3] Aunión, J. A. (2014, December 29). El saber ya no cabe en el campus. Retrieved from http://elpais.com

[4] Blázquez, S. (2014, October 26). Cursos masivos, gratuitos y de prestigio. Retrieved from http://elpais.com

[5] Calzada-Prado, J. (2012). *Recursos educativos abiertos (OER)*. Virtual handout for the subject "Free knowledge and learning on the web".

[6] Champaign, J., Colvin, K. F., Liu, A., Fredericks, C., Seaton, D., & Pritchard, D. E. (2014). Correlating skill and improvement in 2 MOOCs with a student's time on tasks. In *Proceedings of the first ACM conference on learning @ Scale conference* (pp. 11–20). L@S '14. Atlanta, Georgia, USA: ACM. doi:10.1145/2556325.2566250

[7] Committee, L. T. S. (2002). Draft Standard for Learning Object Metadata. IEEE Standard 1484.12.1.

[8] Cormier, D. (2010, December 8). What is a MOOC? [Video]. Retrieved from https://www.youtube.com/watch?v=eW3gMGqcZQc

[9] Cress, U., Dimitrova, V., & Specht, M. (2009). *Learning in the synergy of multiple disciplines: 4th European Conference on Technology Enhanced Learning, EC-TEL 2009 Nice, France, September 29–October 2, 2009 proceedings*. LNCS sublibrary: Programming and software engineering. Springer. Retrieved from http://link.springer.com/content/pdf/10.1007/978-3-642-04636-0.pdf

[10] Cuban, L. (1986). *Teachers and machines: the classroom use of technology since 1920*. Teachers College Press. Retrieved from http://books.google.es/books?id=uQeEn1vEUSQC

[11] Davidson, C. (2013, September 27). What was the first MOOC? [Blog post]. Retrieved from http://www.hastac.org/blogs/cathy-davidson/2013/09/27/what-was-first-mooc

[12] Downes, S. (1998). The future of online learning. *Online Journal of Distance Learning Administration*, *1*(3).

[13] Downes, S. (2013, September 12). What are cultures of learning? [Slideshow]. Retrieved from http://www.downes.ca/presentation/326

[14] Fimognari, A. (2014, April 30). 25 libraries for graphs and charts [Blog post]. Retrieved from http://blog.kurtosys.com/25-libraries-graphs-charts

[15] Foundation, D. S. (Ed.). (2014). *Django documentation. Release 1.4.x*. Retrieved from https://docs.djangoproject.com/en/1.4

[16] Franklin, S. (2013, July 1). Understanding Python decorators in 12 easy steps! [Blog post]. Retrieved from http://simeonfranklin.com/blog/2012/jul/1/python-decorators-in-12-steps

[17] Google (Ed.). (2014, June). *Google Charts documentation*. Retrieved from https://developers.google.com/chart

[18] Goral, T. (2013, July). SPOCs may provide what MOOCs can't. Retrieved from http://www.universitybusiness.com/article/spocs-may-provide-what-moocs-can't

[19] Govaerts, S., Verbert, K., Klerkx, J., & Duval, E. (2010). Visualizing activities for self-reflection and awareness. In *Advances in web-based learning - icwl 2010* (Vol. 6483, pp. 91–100). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-17407-0_10

[20] Ho, A. D., Reich, J., Nesterko, S. O., Seaton, D. T., Mullaney, T., Waldo, J., & Chuang, I. (2014, January 21). *HarvardX and MITx: the first year of open online courses, fall 2012-summer 2013*. HarvardX and MITx. doi:http://dx.doi.org/10.2139/ssrn.2381263

[21] Houston, K. (2012, May 10). [Insert technology here] will revolutionize education [Blog post]. Retrieved from http://www.kevinhouston.net/blog/2012/05/insert-technology-here-will-revolutionize-education

[22] jQuery Foundation, T. (Ed.). (2014). *How jQuery works*. Retrieved from http://learn.jquery.com/about-jquery/how-jquery-works

[23] Kay, J., Reimann, P., Diebold, E., & Kummerfeld, B. (2013, May). Moocs: so many learners, so much potential... *Intelligent Systems, IEEE*, *28*(3), 70–77. doi:10.1109/MIS.2013.66

[24] Kim, J. [Joshua]. (2015, 13 1). The global future of education at 5.5 inches [Blog post]. Retrieved from https://www.insidehighered.com/blogs/technology-and-learning/global-future-education-55-inches

[25] Kim, J. [Juho], Guo, P. J., Seaton, D. T., Mitros, P., Gajos, K. Z., & Miller, R. C. (2014). Understanding in-video dropouts and interaction peaks in online lecture videos. In *Proceedings of the first ACM conference on learning @ Scale conference* (pp. 31–40). L@S'14. Atlanta, Georgia, USA: ACM. doi:10.1145/2556325.2566237

[26] de-la-Fuente-Valentin, L., Burgos, D., & Gonzalez Crespo, R. (2014, July). A4Learning – a case study to improve the user performance: Alumni Alike Activity Analytics to self-assess personal progress. In *Advanced learning technologies (ICALT), 2014 IEEE 14th International Conference on* (pp. 360–362). Athens: IEEE. doi:10.1109/ICALT.2014.107

[27] Larson, W. (2007, December 11). Two-faced Django part 5: JQuery Ajax [Blog post]. Retrieved from http://lethain.com/two-faced-django-part-5-jquery-ajax

[28] Leony, D., Muñoz-Merino, P. J., Pardo, A., & Kloos, C. D. (2013). Provision of awareness of learners' emotions through visualizations in a computer interaction-based environment. *Expert Systems with Applications*, *40*(13), 5093–5100. doi:http://dx.doi.org/10.1016/j.eswa.2013.03.030

[29] Leony, D., Pardo, A., de la Fuente Valentín, L., de Castro, D. S., & Kloos, C. D. (2012). Glass: a learning analytics visualization tool. In *Proceedings of the 2nd international conference on learning analytics and knowledge* (pp. 162–163). LAK '12. Vancouver, British Columbia, Canada: ACM. doi:10.1145/2330601.2330642

[30] Magro-Mazo, C. (2014, September 23). Moocs: sobrevendidos e infrautilizados [Slideshow]. Retrieved from http://es.slideshare.net/carlosmagro/moocs-unesco-uciiim

[31] Marciel, M., Michelinakis, F., Fanou, R., & Muñoz-Merino, P. J. (2013, September). Enhancements to Google Course Builder: assessments visualisation, YouTube events collector and dummy data generator. Retrieved from http://eprints.networks.imdea.org/593/1/sintice2013%5C_11%5C_Marciel.pdf

[32] Martin, F. G. (2012, August). Will Massive Open Online Courses change how we teach? Sharing recent experiences with an online course. *Communications of the ACM*, *55*(8), 26–28. doi:10.1145/2240236.2240246

[33] Mazza, R. & Dimitrova, V. (2004). Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th international World Wide Web conference on alternate track papers and posters* (pp. 154–161). WWW Alt. '04. New York, NY, USA: ACM. doi:10.1145/1013367.1013393

[34] McCulloch, R. & Rothschildneed, L. P. (2014, September). MOOCs: an inside view. *Notices of the American Mathematical Society*, *61*(8). doi:http://dx.doi.org/10.1090/noti1147

[35] Muñoz-Merino, P. J., Kloos, C. D., Wolpers, M., Friedrich, M., & Muñoz-Organero, M. (2010). An approach for the personalization of exercises based on contextualized attention metadata and semantic web technologies. In *Proceedings of the 2010 10th IEEE international conference on advanced learning technologies* (pp. 89–91). ICALT '10. Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICALT.2010.32

[36] Muñoz-Merino, P. J., Ruipérez-Valiente, J. A., Alario-Hoyos, C., Pérez-Sanagustín, M., & Kloos, C. D. (2014, October). Precise effectiveness strategy for analyzing the effectiveness of students with educational resources and activities in

MOOCs. *Computers in Human Behavior*. doi:http://dx.doi.org/10.1016/j.chb.2014.10.003

[37] Muñoz-Merino, P. J., Valiente, J. A. R., & Kloos, C. D. (2013). Inferring higher level learning information from low level data for the Khan Academy platform. In *Proceedings of the third international conference on learning analytics and knowledge* (pp. 112–116). LAK '13. Leuven, Belgium: ACM. doi:10.1145/2460296.2460318

[38] Papert, S. (1993). *The children's machine: rethinking school in the age of the computer*. Basic Books.

[39] Parry, M. (2010, August 29). Online, bigger classes may be better classes. Retrieved from chronicle.com

[40] Rodríguez, C. O. (2012). MOOCs and the AI-Stanford like courses: two successful and distinct course formats for Massive Open Online Courses. *European Journal of Open, Distance and E-Learning*. Retrieved from http://eric.ed.gov/?id=EJ982976

[41] Ruipérez-Valiente, J. A. (2013). *Diseño e implementación de un módulo de analítica de aprendizaje en la plataforma Khan Academy* (Master's thesis, Universidad Carlos III de Madrid).

[42] Ruipérez-Valiente, J. A., Muñoz-Merino, P. J., & Kloos, C. D. (2013). An architecture for extending the learning analytics support in the Khan Academy framework. In *Proceedings of the first international conference on technological ecosystem for enhancing multiculturality* (pp. 277–284). ACM.

[43] Ruipérez-Valiente, J. A., Muñoz-Merino, P. J., Leony, D., & Kloos, C. D. (2014). ALAS-KA: a learning analytics extension for better understanding the learning process in the Khan Academy platform. *Computers in Human Behavior*. doi:http://dx.doi.org/10.1016/j.chb.2014.07.002

[44] Santofimia-Ruiz, J., Pijeira-Díaz, H. J., Ruipérez-Valiente, J. A., Muñoz-Merino, P. J., & Delgado-Kloos, C. (2014). Towards the development of a learning analytics module in Open edX. In *TEEM '14: Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 299–306). Salamanca, Spain: ACM. doi:10.1145/2669711.2669914

[45] Scolari, C. A., Bonito, I. D., & Masanet, M. J. (2014, June). *#UPF2020 Designing the university of the future*. Universitat Pompeu Fabra. Barcelona. Retrieved from http://www.upf.edu/cquid/_pdf/xUPF2020_ENG.pdf

[46] Seaton, D. T. [Daniel T.], Bergner, Y., Chuang, I., Mitros, P., & Pritchard, D. E. (2013). Towards real-time analytics in MOOCs. In *IWTA@LAK'13* (Vol. 985). CEUR Workshop Proceedings. CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-985/paper3.pdf

[47] Seaton, D. T. [Daniel T], Rodenius, A. J., Coleman, C. A., Pritchard, D. E., & Chuang, I. (2013). Analysis of video use in edX courses. *AIED 2013 Workshops Proceedings Volume*, 57. Retrieved from http://people.csail.mit.edu/zp/moocshop2013/presentation_10.pdf

[48] Selingo, J. J. (2014, October 29). Demystifying the MOOC. Retrieved from http://www.nytimes.com/

[49] Shiu, P. K. (2011, January 3). Django database migration tool: south, explained [Blog post]. Retrieved from http://www.djangopro.com/2011/01/django-database-migration-tool-south-explained/

[50] Shum, S. B. (2012, November). *Learning analytics policy brief*. Moscow: UNESCO. Retrieved from http://iite.unesco.org/pics/publications/en/files/3214711.pdf

[51] Siemens, G. [G.] & Tittenberger, P. (2009). *Handbook of emerging technologies for learning*. Retrieved from http://elearnspace.org/Articles/HETL.pdf

[52] Siemens, G. [George]. (2012, July 25). MOOCs are really a platform [Blog post]. Retrieved from http://www.elearnspace.org/blog/2012/07/25/moocs-are-really-a-platform

[53] Torres-Menárguez, A. (2014, November 26). El segundo 'boom' de la enseñanza 'online'. Retrieved from http://elpais.com

[54] UNESCO. (2002, July). *Forum on the impact of Open Courseware for higher education in developing countries*. UNESCO. Paris. doi:CI-2002/CONF.803/CLD.1

[55] Vázquez, K. (2014, October 9). ¿Qué fue de la revolución MOOC? Retrieved from http://elpais.com

[56] W3Schools (Ed.). (2014). *AJAX tutorial*. Retrieved from http://www.w3schools.com/ajax

[57] Wang, L. (2014, November 19). Product roadmap [Web page]. Retrieved from https://openedx.atlassian.net/wiki/display/PROD/Product+Roadmap

[58] 50 JavaScript Libraries for Charts and Graphs [Blog post]. (n.d.). Retrieved from http://techslides.com/50-javascript-charting-and-graphics-libraries

[59] *Edx developer stack*. (2014). Retrieved from https://github.com/edx/configuration/wiki/edX-Developer-Stack

[60] *Mako documentation*. (2014). Retrieved from http://docs.makotemplates.org/en/latest

[61] *The MongoDB 2.X Manual*. (2014). Retrieved from http://docs.mongodb.org/manual

[62] Using edX Insights [Web site]. (2014). Retrieved from http://edx-insights.readthedocs.org/en/latest/index.html

[63] *Vagrant: getting started*. (2014). Retrieved from https://docs.vagrantup.com/v2/getting-started/

[64] Zhang, H., Almeroth, K., Knight, A., Bulger, M., & Mayer, R. (2010, July). Moodog: tracking students' online learning activities. *Journal of Interactive Learning Research*, *21*(3), 407–429.

Chapter A

Appendix

Character or expression	Meaning
^	Starting position within the string.
$	Ending position within the string.
(?P<*name*>*regexp*)	Substring matched by *regexp* is accessible via the name *name*.
—	Logic OR, alternatives.
()	Grouping, scope and precedence of operators.
?	Quantifier: zero or one of the preceding element.
*	Quantifier: zero or more of the preceding element.
+	Quantifier: one or one of the preceding element.
m,n	Quantifier: preceding element minimum m times and maximum n.
.	Any single character.
[]	Single character contained within the brackets.
[^]	Single character not contained within the brackets.
\d	Digit. Same as [0-9].

Table A.1: Some regular expression basics.

SQL	MongoDB
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents and linking
primary key: unique column(s)	primary key: the _id field
WHERE	match
GROUP BY	group
HAVING	match
SELECT	project
ORDER BY	sort
LIMIT	limit
SUM()	sum
COUNT()	sum

Table A.2: Equivalence between SQL and MongoDB terms/instructions [61].